DECOUVREZ LE ZX 81 et le Timex Sinclair 1000



D. Hergert

SYBEX

DECOUVREZ LE

ZX 81 et le Timex Sinclair 1000

DECOUVREZ LE

ZX 81 et le Timex Sinclair 1000

Douglas Hergert

SYBEX

Paris · Berkeley · Düsseldorf

Couverture **Daniel Le Noury**
Traduction française **Jean-Pierre Loison**

ISBN 2-902414-56-0.

A Mme Berthe Badji

Table des matières

Introduction

L'ENTRÉE DANS L'ÈRE DE L'ORDINATEUR

Vous venez de rentrer chez vous avec votre nouvel ordinateur : le Sinclair ZX81. C'est un moment d'excitation et d'enthousiasme; vous êtes enfin entré dans l'ère de l'ordinateur. Quand vos amis parleront de leur ordinateur personnel, vous ne resterez plus dans une tranquille ignorance. Maintenant vous faites partie du club. Vous êtes heureux de votre modeste investissement, comparé aux dizaines de milliers de Francs que d'autres ont dépensés pour leur ordinateur.

Vous branchez votre ordinateur, installez le cordon TV, et remarquez avec beaucoup de satisfaction que votre ordinateur fait ce que vous attendiez.

A ce stade, deux scénarios très différents sont susceptibles de décrire la suite de l'expérience. Le premier scénario se termine en désillusion et regrets; il n'est absolument pas satisfaisant. Le deuxième, dont vous avez raison de souhaiter la réalisation, est celui à l'intérieur duquel ce livre vous guidera. Considérons ces deux scénarios.

Dans le premier, vous commencez à taper sur le clavier et à regarder ce qui se passe sur l'écran; au début, vous vous émerveillez du pouvoir que vous croyez avoir sur le fonctionnement de l'ordinateur. Mais petit à petit vous vous apercevez que vous ne *comprenez* pas réellement ce qui se passe. En fait,

l'ordinateur travaille chaque fois que vous appuyez sur le clavier, mais que fait-il ? Vous commencez à réaliser que peut-être le travail sera plus long que vous ne l'aviez prévu. Tant pis, pensez-vous, le manuel expliquera tout cela. Aussi ouvrez-vous le petit manuel à dos-spirale livré avec votre ordinateur, et commencez-vous à lire. Au bout d'un paragraphe ou deux, cela commence à ne pas vous sembler facile. Après un certain nombre de pages, vous vous sentez complètement perdu. D'une manière ou d'une autre, vous avez l'impression que ce livre s'adresse à des gens en sachant déjà plus que vous. Vous pressentez cependant qu'il comporte quantité d'informations utiles à quelqu'un n'ayant qu'une petite expérience – ce qui est votre cas – mais comment faire pour démarrer ? Vous retournez à l'ordinateur, appuyez avec optimisme sur quelques touches, en vous disant que vous allez y arriver bientôt. Mais non !!... Une partie de votre enthousiasme du départ est perdue, s'est transformée en un sentiment inattendu de désarroi et de déception. Vous débranchez l'ordinateur, le rangez dans un placard, et sortez tondre la pelouse... Deux ou trois mois plus tard, vous apercevez l'ordinateur, toujours dans son placard, avec sur le clavier une fine couche de poussière. Quand vous discutez avec les gens, vous aimez dire que vous *possédez* votre propre ordinateur, mais hélas, vous vous arrangez toujours pour changer de sujet quand vos amis vous posent des questions précises...

Dans le deuxième scénario, vous abordez votre ordinateur d'une façon plus méthodique. Vous êtes parfaitement conscient d'avoir beaucoup à apprendre sur votre nouvel ordinateur, et plus vous apprenez, plus vous êtes enthousiaste. Vous commencez par maîtriser le clavier du ZX81. Vous apprenez à introduire des commandes, et vous découvrez la signification de chacune d'entre elles. Vous tapez un programme BASIC en entier. (Le BASIC est le langage disponible sur le ZX81.) Vous êtes impressionné par l'aspect intéressant des résultats. L'expérience devient plus fascinante à chaque fois que vous vous asseyez devant votre ordinateur; vous étudiez les graphismes, les calculs, la manipulation de chaînes de caractères... et avant même de vous en rendre compte vous êtes devenu un programmeur expérimenté en BASIC qui ne s'étonne pas seulement lui-même, mais qui surprend aussi toute sa famille et ses amis. Au bout de plusieurs semaines, vous

réalisez que vous n'êtes plus seulement un possesseur d'ordinateur, vous êtes devenu un *utilisateur* d'ordinateur, ce qui est beaucoup plus important.

Ce livre a été écrit pour que vous suiviez le deuxième scénario. Si vous ne connaissez rien aux ordinateurs, si vous êtes totalement perdu dans ce langage informatique que les autres semblent saisir automatiquement, alors ce livre est fait pour vous. Il commence au tout début de l'histoire, puis vous guide soigneusement le long des étapes qu'il vous faut franchir pour devenir un utilisateur d'ordinateur satisfait.

Le Chapitre 1 – *Présentation des personnages* – décrit le *matériel (Hardware)* – les appareils dont vous avez besoin pour obtenir le meilleur de votre ordinateur. Il vous donne également soigneusement la marche à suivre pour connecter les autres équipements à votre ordinateur. Enfin, ce chapitre présente ce que vous avez à faire pour que l'ordinateur puisse travailler pour vous, et les techniques que vous apprendrez à mettre en œuvre en utilisant votre ordinateur.

Le Chapitre 2 – *Le Premier Acte : Introduire votre programme* – est une introduction au *logiciel (Software)* – les instructions et les programmes que vous devez préparer pour guider votre ordinateur dans des tâches spécifiques. Vous commencerez par apprendre exactement de quelle façon vous pouvez utiliser le clavier du ZX81 pour communiquer avec votre ordinateur. Vous introduirez un programme en entier, le ferez fonctionner, et ce faisant, commencerez à entrevoir la puissance du langage de programmation BASIC. Enfin, vous apprendrez comment *sauvegarder* un programme de façon permanente sur cassette pour pouvoir réutiliser par la suite ce programme. Ce chapitre se termine par un guide de l'acheteur de *logiciel,* et par l'exposé du pour et du contre entre l'achat de programmes tout faits et l'écriture par vous-même de vos propres programmes.

Le Chapitre 3 – *L'affaire se corse* – commence par un rapide et facile cours de BASIC. Vous apprendrez

comment dire à l'ordinateur de répéter certaines tâches; comment prendre des décisions à partir des informations dont il dispose; comment afficher à l'écran les résultats afin de les étudier. Vous apprendrez aussi que le ZX81 dispose de *deux* modes graphiques, et uniquement pour votre plaisir, vous verrez un programme complet qui vous permettra de tracer des dessins sur l'écran TV.

Le Chapitre 4 – *Recette numéro 5* – concerne tout ce qui a trait aux nombres et aux calculs sur votre ordinateur. Vous découvrirez comment stocker de grandes quantités de données numériques dans votre ordinateur de façon commode et efficace. Vous verrez de quelle manière transformer votre ordinateur en un "super-calculateur". Enfin, le principal programme de ce chapitre vous montrera comment créer des histogrammes à partir de n'importe quelles informations numériques.

Le Chapitre 5 – *Des mots, des mots, des mots* – décrit la façon dont le BASIC peut manipuler les *chaînes de caractères* – c'est-à-dire les informations non numériques. Le chapitre comporte un programme amusant qui pourra vous aider dans les moments difficiles de la vie, quand vous ne savez quelle décision prendre.

L'Appendice A décrit le vocabulaire du BASIC – c'est-à-dire tous les mots apparaissant sur votre clavier. Vous pourrez utiliser cet appendice comme outil de référence lors de l'écriture de vos programmes.

L'Appendice B résume les codes-erreurs du ZX81.

Ce livre est conçu de telle façon que les informations introduites dans un chapitre sont utilisées dans les chapitres suivants; chaque chapitre est construit à partir des chapitres précédents. Pour utiliser avec succès ce livre, il faut travailler avec l'ordinateur à portée de main, en expérimentant chaque nouvelle instruction comme indiqué, en introduisant et faisant fonctionner tous les programmes au fur et à mesure qu'on les rencontre. Si vous consacrez le temps qu'il faut à maîtriser chaque étape avant de

passer à la suivante, une fois arrivé à la fin de ce livre, vous aurez acquis une parfaite connaissance des possibilités de votre ordinateur et des meilleurs moyens de les utiliser.

La chose la plus importante à emporter chez vous, en sortant du magasin avec votre ordinateur ZX81, c'est énormément d'enthousiasme pour votre nouvel ordinateur. Ne perdez pas cet enthousiasme. Laissez ce livre vous apprendre ce qu'il vous faut savoir pour tirer de votre nouvel ordinateur le meilleur de lui-même.

Remarque à l'intention des utilisateurs du Timex Sinclair 1000

Vous pouvez utiliser ce livre en tant qu'introduction à la pratique de votre nouvel ordinateur. Le Timex Sinclair 1000 possède les mêmes commandes, le même clavier (à deux différences près) et utilise les mêmes équipements que le ZX81. La seule différence significative entre ces deux ordinateurs est la taille mémoire. Alors que le ZX81 ne dispose que de 1 K de mémoire, le Timex Sinclair 1000 en a 2. Plusieurs des programmes de ce livre nécessitent 2 K de mémoire; cependant, si vous décidez d'augmenter la mémoire du ZX81 avec le module d'extension de 16 K de mémoire supplémentaire, vous pourrez non seulement faire fonctionner ces programmes sur votre ordinateur, mais aussi des programmes beaucoup plus longs.

Chapitre 1

PRÉSENTATION DES PERSONNAGES

INTRODUCTION

Comme vous le savez déjà certainement, le microordinateur ZX81 – une machine amusante et puissante bien que présentée dans un petit boîtier de plastique noir brillant – est très simple à utiliser. Pour le faire effectivement fonctionner, vous devez le relier à d'autres équipements. Un certain nombre d'entre eux sont courants : un poste de télévision, et un lecteur-enregistreur à cassettes. D'autres sont spécifiques à l'ordinateur ZX81 : par exemple, l'imprimante et le module de mémoire supplémentaire. La plupart de ces équipements sont chargés de deux sortes de travaux : envoyer des informations vers, ou bien recevoir des informations depuis votre ordinateur. On appelle respectivement ces deux tâches les *entrées* et les *sorties (input* et *output).*

Dans ce premier chapitre nous examinerons rapidement le microordinateur et ses accessoires – obligatoires ou optionnels –. Nous verrons ce que font ces différents accessoires et nous apprendrons à les relier convenablement à l'ordinateur. Nous examinerons également les différents rôles que vous, l'utilisateur, aurez à jouer. A la fin de ce chapitre, vous serez prêt à commencer le travail avec votre ordinateur.

LE PREMIER RÔLE : LE ZX81

Jetons un coup d'œil à l'ordinateur. C'est une bonne chose que de passer quelque temps à l'examiner et de noter un certain nombre de ses caractéristiques avant de commencer réellement à travailler avec lui. Les Figures 1.1 à 1.3 présentent l'ordinateur sous trois angles différents; considérons-les tour à tour.

Sur le devant de l'ordinateur se trouve le clavier. Grâce à lui vous pouvez introduire des commandes et fournir des informations à l'ordinateur. Ainsi, le clavier de l'ordinateur est sa principale unité d'entrée. Au début, cela peut vous surprendre qu'une surface plate en plastique puisse réellement servir de clavier. Mais vous découvrirez bientôt que les touches sont *sensitives.* Cela signifie qu'une fois l'ordinateur sous tension, une



Figure 1.1 : Le clavier du ZX81

légère pression du doigt sur l'une des touches provoquera l'envoi d'une information du clavier vers l'ordinateur.

Les 26 lettres de l'alphabet, et les dix chiffres, disposés dans le même ordre que sur une machine à écrire de type QWERTY occupent la majeure partie du clavier. Mais à côté des lettres et des chiffres, le clavier comporte également un ensemble de mots, de symboles, et de signes graphiques. Les mots sont les commandes et les instructions que vous pouvez fournir à votre ordinateur dans le but de lui demander de faire quelque chose. Vous apprendrez la signification de tous ces mots en lisant ce livre et en commençant à utiliser votre ordinateur.

Pourquoi avoir mis ces mots de commande directement sur le clavier ? C'est simple. Au lieu de devoir les taper caractère par caractère, vous n'avez qu'à appuyer sur une seule touche pour envoyer à votre ordinateur une commande ou une instruction. Ceci peut ne pas vous sembler particulièrement commode de prime abord; vous passerez peut-être un peu plus de temps au départ à chercher les mots de commande sur le clavier, que si vous les aviez tapés caractère par caractère. Mais avec la pratique, vous serez familiarisé avec la disposition du clavier, et vos doigts iront directement sur les touches que vous souhaitez frapper.

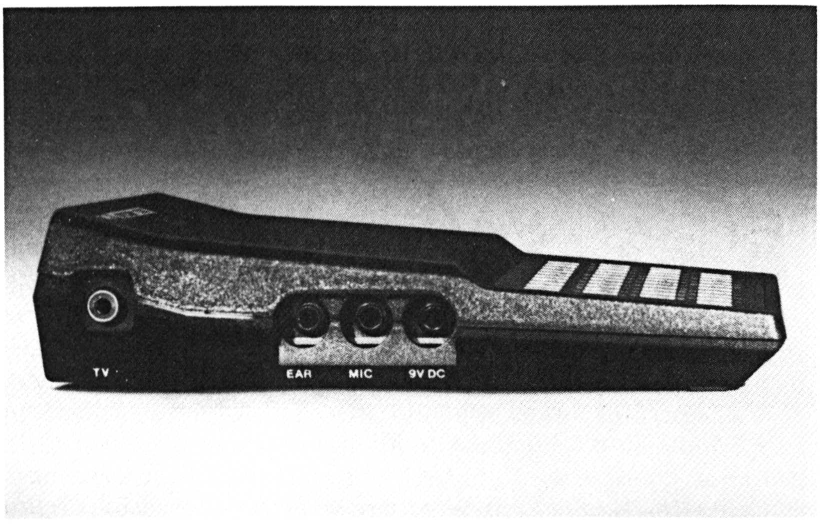


Figure 1.2 : Le côté gauche de votre ordinateur

Naturellement, il est difficile de taper avec les deux mains sur ce clavier de la même façon que sur un clavier de machine à écrire, mais vous serez surpris de l'efficacité que vous atteindrez au bout de quelques jours d'utilisation seulement.

Puisque la plupart des touches ont jusqu'à cinq symboles ou mots, un système est nécessaire pour *basculer* l'utilisation du clavier d'une catégorie de symboles ou de mots à une autre. Il y a plusieurs sortes de modes de saisie avec le clavier du ZX81, et nous les maîtriserons tous au Chapitre 2. De même que les positions des mots de commandes, le passage d'un mode à un autre deviendra naturel pour vous en peu de temps.

Maintenant, regardez le côté gauche de votre ordinateur (Figure 1.2); vous voyez 4 prises. Le rôle de chacune d'entre elles est indiqué : la première sert à relier l'ordinateur au poste de télévision; les deux du milieu sont prévues pour le lecteur-enregistreur à cassettes; et la dernière, au-dessous de laquelle est écrit "9V DC", sert à l'alimentation électrique.

Avec votre ordinateur, vous avez les prises, cordons et autres équipements matériels dont vous aurez besoin pour relier ensemble tous les éléments de votre système. Faisons un rapide inventaire de ces éléments.

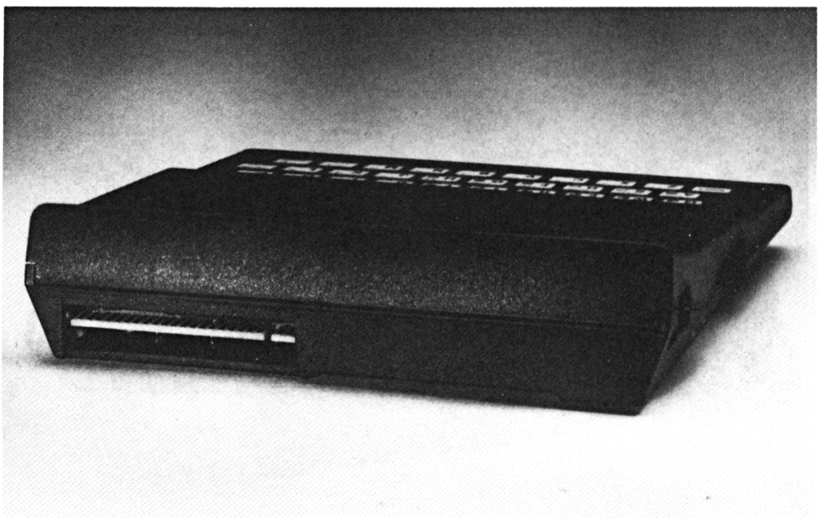


Figure 1.3 : L'arrière de l'ordinateur

Un cordon d'environ 1 mètre de long, avec une grosse fiche à chaque extrémité assure la connexion avec le téléviseur. Vous pouvez dès maintenant brancher l'une des fiches dans la prise de l'ordinateur marquée "TV". Nous verrons bientôt comment brancher l'autre fiche sur le téléviseur. En fait, aucune des fiches électriques de l'ordinateur ne présente de danger, même si le courant est mis.

Vous devez également avoir un petit cordon muni d'une paire de fiches à chaque extrémité qui servira pour le lecteur-enregistreur à cassettes. Mettez-le de côté pour l'instant (mais ne le perdez pas); nous examinerons en détail la liaison avec le magnétophone au Chapitre 2.

Enfin vous trouverez un troisième cordon muni d'une fiche jack, d'un adaptateur de courant, et d'une prise secteur. Cette prise est prévue pour n'importe quelle prise 220 V, avec une rallonge si nécessaire, et la fiche jack pour la prise 9V DC située sur le côté de votre ordinateur. Vous pouvez la brancher maintenant si vous voulez l'essayer. Dès que vous l'aurez fait l'ordinateur sera prêt à fonctionner. (Malheureusement, vous n'aurez aucune idée de ce que fait l'ordinateur si vous ne reliez pas le cordon TV.) C'est une bonne idée de débrancher l'adaptateur chaque fois que vous cessez d'utiliser l'ordinateur pour un certain temps.

A l'arrière de votre ordinateur (comme vous le voyez à la Figure 1.3) se trouve une ouverture dans le boîtier d'environ 6 cm de longueur et 1 cm de largeur. Cette ouverture fait apparaître le bord du circuit imprimé, sur lequel vous connecterez la mémoire supplémentaire ou l'imprimante, si vous décidez d'acquérir l'un ou l'autre de ces deux accessoires optionnels. Nous en parlerons plus loin dans ce chapitre. (Si vous disposez déjà de l'un d'entre eux ou des deux, voici une mise en garde : ne les branchez ni ne les débranchez avec l'ordinateur sous tension.)

Maintenant que nous avons examiné l'extérieur de l'ordinateur, qu'y a-t-il à l'intérieur – quel est le *contenu* de la boîte noire ? En quoi avez-vous besoin de savoir quoi que ce soit sur l'organisation du ZX81 pour devenir un utilisateur accompli d'ordinateur ? Sans doute avez-vous entendu parler, ou lu des articles au sujet des progrès réalisés dans la technologie des circuits intégrés (les *puces*) à l'origine de la révolution de l'ordinateur personnel. Vous

savez peut-être que la puce qui "est le cerveau" de votre ZX81 s'appelle le microprocesseur Z80. De quelle manière cette information peut-elle vous être utile pour jouer avec votre ordinateur aux échecs, ou l'utiliser pour votre budget familial ?

La réponse, au cas où vous ne l'auriez pas devinée, est "en aucune façon". Tout ce que vous avez besoin de savoir apparaît à l'extérieur : l'ensemble des commandes du clavier, et les informations affichées sur votre écran de télévision. En ce qui vous concerne, la structure interne de votre ordinateur pourrait être résumée aussi bizarrement que le fait Ian Frazier dans le passage qui suit, extrait de sa courte nouvelle, *The Killion :*

> *« Les gens qui ne sont pas familiarisés avec les ordinateurs trouvent souvent utile d'y penser comme à des choses compliquées, d'assez grande taille. Les dimensions des ordinateurs vont de celle d'un petit seau à glace de restaurant à celle d'un complexe de loisirs tel qu'à New Jersey's Meadowlands, parking compris. A l'intérieur de l'ordinateur, il y aura un petit fil électrique rouge connecté à l'une de ses extrémités à un terminal; à l'autre extrémité il y aura un autre terminal. Et il y aura aussi un fil bleu relié à l'un des terminaux, et puis un fil vert, et un fil jaune, puis un fil orange, puis un fil rose, et ainsi de suite. »*

Ou bien vous pouvez croire à l'histoire de l'attelage de puces très bien dressées... Le paradoxe est le suivant : votre nouvel ordinateur a été conçu de telle façon que vous n'avez pas besoin de connaître ni son architecture ni son fonctionnement interne. (Remarquez que le boîtier en plastique noir qui le contient n'est pas prévu pour être ouvert.) Il en est de même pour beaucoup d'autres appareils de votre maison – du téléphone au moulin à café – vous n'avez pas besoin d'être expert en technologie pour pouvoir faire fonctionner une machine de façon totalement satisfaisante.

Ceci étant dit, il faut préciser que de nombreux livres et journaux sont disponibles pour vous apprendre tout ce que vous voudrez connaître de la technologie des microprocesseurs et des ordinateurs. En travaillant avec votre nouvel ordinateur, et en connaissant mieux la manière de l'utiliser, il se peut que vous vous aperceviez que ce sujet vous intéresse, et que vous vouliez en savoir davantage. Mais vous n'avez pas besoin de ces

connaissances pour utiliser votre ordinateur; c'est à vous de décider ce que vous voulez apprendre – et jusqu'où pousser cette étude.

Enfin, en tant que nouvel utilisateur d'un ordinateur, vous serez peut-être quelque peu inquiet quant à ce qui pourrait l'endommager. Ne vous en faites pas. Votre ZX81 ne craint pas grand chose. Et c'est seulement à l'occasion que vous lirez les précautions d'emploi. (Prenez simplement un certain nombre de précautions de simple bon sens : ne laissez pas le chien mordiller votre ordinateur; n'utilisez pas votre ordinateur quand vous êtes dans votre bain; ne le jetez pas par la fenêtre de votre appartement au dixième étage; etc.) Mais vous serez content de savoir qu'avec une utilisation normale, vous ne pouvez pas réellement l'abîmer. Assurément, rien de ce que vous taperez sur son clavier ne pourra en aucun cas l'endommager physiquement. Vous pouvez faire des erreurs qui se traduiront temporairement par de mauvaises réponses ou des pertes d'informations; mais une fois que vous vous en serez aperçu vous aurez toute chance de pouvoir les corriger. Les erreurs font partie de l'apprentissage; l'ordinateur est toujours indulgent.

Regardons l'unité principale de l'ordinateur et découvrons les connexions qu'il faut faire.

LES SECONDS RÔLES

Le poste de télévision

Vous devez pouvoir relier n'importe quel téléviseur à votre ZX81. Un petit téléviseur portable pouvant être posé sur un bureau serait parfait, mais n'importe quel autre genre de téléviseur conviendra.

Le rôle du téléviseur est de fournir toutes les informations dont vous avez besoin sur le fonctionnement de l'ordinateur. Quand vous introduisez des commandes au clavier, vous verrez ces commandes s'afficher sur l'écran avant même de dire à l'ordinateur de les exécuter. Souvent vous écrirez des suites de commandes que vous ne voudrez soumettre à l'ordinateur que

d'un seul coup, pour qu'il les exécute séquentiellement. De telles séries de commandes sont appelées des *programmes.* Avant de dire à l'ordinateur d'exécuter les commandes contenues dans un programme – et que l'on nomme *instructions* – vous pouvez afficher le programme lui-même, ligne par ligne, paragraphe par paragraphe, à l'écran. Le programme peut rester affiché sur l'écran aussi longtemps que vous le souhaitez afin de pouvoir l'étudier et de vous assurer qu'il est correct. Un tel affichage d'un programme se nomme la *liste* du programme. Si vous trouvez une erreur dans votre programme, vous pouvez *éditer* la partie du programme qui est incorrecte; les étapes de l'*édition* vous sont affichées à l'écran. Enfin, quand vous *exécutez* votre programme (c'est-à-dire quand vous dites à l'ordinateur d'exécuter les instructions du programme) vous pouvez voir, dans la plupart des cas, les résultats sur l'écran TV. Les résultats peuvent être numériques, sous forme de textes ou encore de dessins, car le ZX81 peut produire deux sortes de graphismes.

Naturellement, l'ordinateur a toujours le contrôle de ce qui apparaît à l'écran. Quand vous aurez l'habitude de lire les informations que l'ordinateur affiche, vous saurez reconnaître même des détails très précis de l'activité de l'ordinateur. Par exemple vous serez capable de dire, à partir de ce qui apparaît sur l'écran, que l'ordinateur fonctionne en utilisant peu d'espace mémoire. L'ordinateur affichera aussi des messages sur l'écran – que vous devrez apprendre à interpréter – et qui vous diront si vous avez fait une faute. Dans la plupart des cas vous saurez exactement où se trouve la faute, et pourquoi l'ordinateur ne peut l'accepter.

Si le clavier est votre moyen de communication avec l'ordinateur, l'écran TV est le principal moyen qu'utilise l'ordinateur pour vous communiquer des informations. Au début vous trouvez qu'il y a plus d'informations sur l'écran que vous n'êtes prêt à en recevoir, mais rapidement vous saurez comprendre le moindre signe ou événement se passant à l'écran, et serez capable de concentrer toute votre attention sur les informations que vous donne l'ordinateur.

De plus, quand vous écrirez un programme, vous aurez beaucoup de choses à dire sur le pourquoi et le comment de ce que l'ordinateur affiche à l'écran. Plusieurs commandes ont pour

objet de dire à l'ordinateur de quelle façon, et à quel endroit de l'écran doivent être affichés nombres, textes et graphismes.

Votre ordinateur est relié au téléviseur par un cordon; regardez à l'arrière de votre téléviseur. Vous trouverez une prise pour antenne extérieure. Si votre téléviseur comporte deux prises d'antennes, utilisez la prise UHF. Attention il vous faut un téléviseur recevant au moins deux chaînes ! Vous n'avez qu'à brancher dans cette prise la deuxième fiche du cordon.

Vous pouvez mettre votre téléviseur sur le deuxième canal, ou mieux le canal 4. Mettez le volume au minimum. Mettez l'ordinateur et le téléviseur sous tension. Pour mettre l'ordinateur sous tension procédez de la façon suivante : branchez la prise secteur de l'adaptateur puis introduisez la fiche jack de l'adaptateur dans la prise marquée 9V DC de votre ordinateur. Si toutes les connexions ont été bien faites, vous devez voir apparaître le caractère K en *vidéo inversée* (c'est-à-dire en blanc sur noir), dans

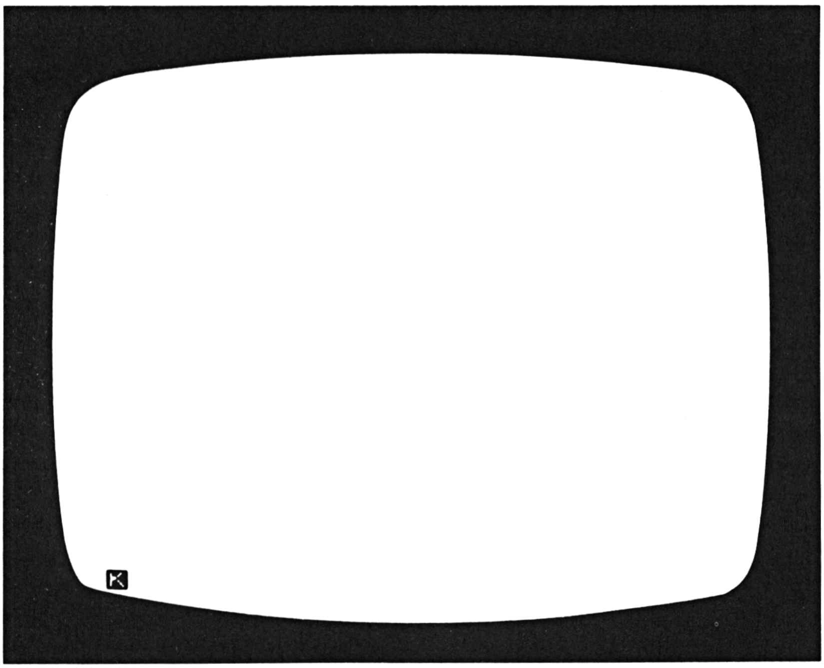


Figure 1.4 : Mise en route de l'ordinateur

le coin en bas à gauche de l'écran, comme vous pouvez le voir à la Figure 1.4. Si l'accord de votre téléviseur est réglable, ajustez-le jusqu'à obtenir ce caractère K. Ce K, signifiant *mot clé (Keyword),* vous indique que l'ordinateur est prêt à recevoir des commandes. Vous pouvez régler la luminosité et le contraste de votre TV à votre convenance. Ces réglages ne sont pas nécessairement les mêmes que pour regarder des émissions. Si l'image ne vous satisfait pas complètement, vous pouvez essayer un autre canal, et voir si le résultat est meilleur.

Si vous n'obtenez pas d'image du tout, vérifiez que les connexions sont bien établies :

- L'adaptateur d'alimentation doit être correctement enfoncé dans une prise (avec du courant !); la fiche mâle jack de l'adaptateur doit être bien enfoncée dans la prise 9V DC sur le côté de l'ordinateur.
- Le cordon de liaison vidéo doit relier l'ordinateur à la prise d'antenne du téléviseur, qui doit être réglé sur une chaîne autre que la première.

Une fois le téléviseur connecté à votre ordinateur, vous êtes prêt à commencer à introduire des commandes et à expérimenter les possibilités du ZX81. En fait, vous pouvez décider de ne pas lui adapter d'autres éléments dans un premier temps. En n'ayant qu'un téléviseur pour vous montrer ce que fait l'ordinateur, vous pouvez apprendre beaucoup de choses.

Cependant, il se peut que, avec un peu plus d'expérience, vous souhaitiez donner à votre ordinateur des possibilités supplémentaires. Vous pourrez lui ajouter de la mémoire supplémentaire afin d'écrire des programmes plus longs. Vous pourrez également vouloir commencer à sauvegarder de façon permanente vos programmes, ou bien vous aurez envie de faire fonctionner sur votre ordinateur des programmes écrits par d'autres personnes que vous. Ou enfin, vous pourrez vouloir mettre sur papier les résultats que vous obtenez. Les paragraphes suivants de ce chapitre décrivent les différents accessoires qui vous permettront de faire tout cela.

La mémoire supplémentaire

L'ordinateur possède deux types de mémoire : la mémoire permanente et la mémoire temporaire; on les appelle aussi *mémoire morte (ROM)* et *mémoire vive (RAM).* La mémoire morte comporte toutes les informations définissant la *personnalité* du ZX81. Ce sont ces informations qui déterminent les commandes que vous pouvez donner à votre ordinateur, et la façon dont il les interprétera. Les informations présentes en *mémoire morte* ne sont jamais perdues, même quand vous mettez l'ordinateur hors tension.

L'autre type de mémoire, la *mémoire vive,* permet de stocker les informations concernant les activités de l'ordinateur, à un moment précis. Par exemple, n'importe quelle commande que vous venez d'introduire, ou n'importe quel programme que vous venez de taper peuvent s'y trouver. Elle garde trace des images que l'ordinateur envoie sur l'écran TV. Elle conserve également les informations que vous avez fournies à l'ordinateur, et les résultats des calculs que vous faites faire à l'ordinateur. La mémoire vive est très temporaire; les informations qui s'y trouvent peuvent être modifiées à tout moment. Il est très important de vous rappeler que lorsque vous débranchez votre ordinateur, toutes les informations qui étaient stockées en mémoire vive sont perdues.

L'unité servant à définir la taille de la mémoire de votre ordinateur est l'*octet.* Un octet est une quantité de mémoire dans laquelle l'ordinateur peut ranger l'information provenant d'une des touches du clavier. Ainsi, si vous introduisez l'une des différentes commandes ou l'un des différents caractères imprimés sur le clavier, l'ordinateur a besoin d'un octet pour en garder trace.

Maintenant, sachez que votre ZX81 dispose de 1 K de mémoire vive (soit 1024 octets). En effet l'unité K (Kilo-octet) vaut 1024 (ou 2 à la puissance 10). Si vous disposez d'un TS 1000 vous avez alors 2 K de mémoire vive. Vous verrez que c'est suffisant pour écrire des programmes courts, mais cependant intéressants et utiles. Si vous utilisez le TS 1000, sa mémoire sera tout juste suffisante pour les programmes décrits dans ce livre, mais si vous avez un ZX81, vous aurez absolument besoin d'un module d'extension mémoire.

De toute façon, vous pouvez souhaiter écrire vous-même de longs programmes ou en utiliser de très longs écrits par d'autres. Vous pouvez trouver des modules d'extension mémoire de 16 K (produits par SINCLAIR ou TIMEX), et d'autres constructeurs proposent des extensions jusqu'à 64 K. La quantité de mémoire dont vous avez besoin dépend de ce que vous voulez faire avec votre ordinateur.

Ces modules s'enfichent à l'arrière de l'ordinateur. (Une fois de plus assurez-vous, avant de le faire, que l'ordinateur est hors tension. C'est indispensable aussi bien pour installer que pour retirer le module d'extension.) Un module n'est indispensable que si vous avez un ZX81; si au contraire vous disposez d'un TS 1000, ses 2 K de mémoire vive seront suffisants pour les programmes de cet ouvrage. Cependant nous en examinerons certains, vendus dans le commerce, qui nécessitent 16 K.

Le lecteur-enregistreur à cassettes

Souvenez-vous que nous avons défini un *programme* comme une suite d'instructions que vous soumettez en une seule fois à l'ordinateur. Quand vous *exécutez* un programme, vous dites à l'ordinateur d'obéir aux instructions, en les prenant les unes après les autres, dans l'ordre que vous avez spécifié. L'autre point dont il faut se souvenir à propos des programmes, c'est qu'ils sont stockés en mémoire *vive,* donc de façon temporaire. Cela signifie que si, travaillant à un programme, l'ordinateur est volontairement ou accidentellement mis hors tension, votre programme sera perdu.

Mais votre ordinateur apporte une bonne solution à ce problème. Vous pouvez stocker un programme en l'enregistrant directement de l'ordinateur sur cassette. Puis ultérieurement, quand vous souhaiterez exécuter à nouveau ce programme, vous repasserez l'enregistrement, et l'ordinateur le rangera à nouveau en mémoire vive. Pour être fiable, cette méthode exige des connexions soigneuses entre l'ordinateur et le lecteur-enregistreur à cassettes; nous verrons cela au Chapitre 2. L'enregistrement des programmes ne nécessite nullement un matériel d'enregistrement coûteux ou sophistiqué. Si vous avez chez vous

un simple magnétophone à cassettes, il y a des chances qu'il fasse l'affaire. Si vous n'en avez pas, vous pouvez éventuellement vouloir faire cet investissement mais achetez-en un bon marché.

L'imprimante

Un autre élément que vous pourrez envisager d'acheter est l'imprimante, conçue pour être utilisée avec le ZX81. De même que le module de mémoire supplémentaire, l'imprimante se connecte à l'arrière de votre ordinateur, et il faut mettre hors tension l'ordinateur avant de l'installer. Si vous avez, *à la fois* une imprimante et un module d'extension mémoire, vous devez installer en premier lieu l'imprimante sur l'ordinateur, puis le module d'extension sur le connecteur de l'imprimante. L'imprimante utilise des rouleaux de papier spécial; le document produit, bien que lisible, est un support médiocre, mais qui "passe assez bien à la photocopie". De toute façon, la possibilité de produire un document écrit à partir de votre ordinateur est déjà quelque chose d'intéressant.

L'ordinateur dispose de plusieurs commandes vous permettant de contrôler l'imprimante directement à partir du clavier. Vous pouvez, par exemple, dire à l'ordinateur de sortir la liste du programme sur papier, ou de copier tout ce qui se trouve à l'écran. Vous pouvez également écrire des programmes avec des résultats sur papier et non à l'écran. Si vous décidez d'acheter une imprimante, vous verrez qu'elle permet très facilement de produire toutes sortes d'impressions à partir de votre ordinateur. (L'Appendice A décrit les commandes qui permettent de travailler avec l'imprimante : COPY, LLIST et LPRINT.)

LE PRODUCTEUR, LE METTEUR EN SCÈNE ET LE SCRIPT : VOUS

Nous allons maintenant nous tourner vers *vous* afin d'examiner les différents rôles que vous allez jouer, en faisant fonctionner pour vous l'ordinateur.

Nous avons vu que l'ordinateur est conçu pour prendre en

compte et exécuter les commandes que vous pouvez lui communiquer en appuyant sur les touches appropriées du clavier. Si vous regardez le clavier, vous verrez que s'y trouvent environ une soixantaine de mots différents compris par l'ordinateur. Ces mots, ainsi que leurs règles d'utilisation, peuvent être considérés comme un langage : un langage *informatique.* Ce langage informatique est beaucoup plus simple que le langage naturel des humains; il n'a qu'un vocabulaire limité, et une syntaxe simple et sans exceptions. Pour cette raison, vous pourrez l'apprendre en beaucoup moins de temps que vous ne mettriez à apprendre une langue telle que l'anglais ou l'allemand. De plus, la plupart des mots de son vocabulaire sont des mots courants du vocabulaire anglais. Ils seront faciles à retenir dans la mesure où vous aurez appris leur signification au cours d'expérimentations.

Il existe de nombreux langages informatiques. Celui qui est installé sur le ZX81, et que l'on trouve sur la plupart des petits ordinateurs s'appelle le BASIC *(Beginner's All-purpose Symbolic Instruction Code).* Vous entendrez parler de plus en plus du BASIC dans les quelques années à venir, car c'est un langage si facile à apprendre que tout le monde l'utilise pour résoudre toutes sortes de problèmes sur ordinateurs. En France, de jeunes enfants commencent à apprendre à écrire des programmes BASIC. Ainsi la programmation en BASIC sera peut-être bientôt aussi répandue, à la maison ou au bureau, que l'utilisation d'une calculatrice de poche.

Le langage BASIC doit sa popularité à ce qu'il peut être utilisé par des non-spécialistes en informatique. Vous serez ainsi vraiment surpris en découvrant la puissance du langage BASIC quand vous utiliserez votre ordinateur. Ce langage simple a réellement beaucoup de points communs avec d'autres langages qu'utilisent les programmeurs professionnels sur des ordinateurs beaucoup plus gros que le ZX81.

Le premier rôle que vous jouerez avec votre ordinateur sera celui de programmeur BASIC. Après avoir imaginé un travail que vous voulez faire faire par l'ordinateur, vous trouverez vous-même intuitivement la façon de l'organiser :

1. Vous le diviserez en une série d'étapes élémentaires que l'ordinateur peut exécuter les unes après les autres. Ces étapes peuvent comprendre des calculs, des tâches répé-

titives, ou même des prises de décisions que vous voulez que l'ordinateur exécute à partir des données dont il dispose.

2. Vous déterminerez le moyen le plus efficace d'exprimer ces différentes tâches sous forme d'instructions BASIC. Si votre programme est long, avec beaucoup d'instructions, vous commencerez probablement par l'écrire sur papier plutôt que de le taper directement.
3. Une fois que vous aurez écrit complètement le programme, vous en introduirez chacune des lignes dans l'ordinateur. Puis vous le testerez pour voir s'il fait exactement le travail pour lequel vous l'avez conçu. Si le programme effectue des calculs numériques, vous pourrez l'essayer avec des jeux de données différents, avant d'être certain qu'il fonctionne correctement.
4. Si les résultats obtenus ne sont pas corrects, vous reviendrez sur les instructions BASIC que vous avez écrites, et vous essaierez de trouver ce qui ne va pas. Souvent la nature de l'anomalie vous montrera directement d'où vient le problème; d'autres fois les erreurs seront moins faciles à détecter, et vous pourrez passer des heures – ou des jours – avant de résoudre le puzzle qu'est votre programme !

Vous verrez votre premier programme BASIC au Chapitre 2. En réalité, avec ce programme, vous prêterez davantage attention à l'utilisation du clavier du ZX81 qu'à la signification des instructions BASIC, mais vous écrirez bientôt vos propres programmes. Remarquez que le processus de développement des programmes, en quatre étapes, décrit ci-dessus, comporte une étape de détection et de correction des erreurs. Il est inévitable qu'un nouveau programme comporte des erreurs au départ. Cela ne doit en aucun cas vous rendre honteux ! Le ZX81 détectera les quelques erreurs que vous ferez en introduisant le programme. D'autres erreurs apparaîtront quand vous exécuterez le programme. Nous verrons que l'ordinateur possède plusieurs moyens pour vous permettre de trouver et de corriger les erreurs de vos programmes.

Quand vous avez acheté votre ordinateur, vous avez certainement remarqué que vous pouviez aussi acheter des programmes –

écrits par d'autres – et conçus pour le ZX81. Ces programmes sont fournis sur cassettes et sont pour la plupart d'un prix abordable. Mais ceci est une nouvelle façon d'utiliser votre ordinateur – en exécutant des programmes écrits par d'autres, pour réaliser des tâches spécifiques. Au Chapitre 2 nous aborderons quelques-uns des avantages et des inconvénients des programmes du commerce (par opposition à "vos programmes personnels"). Pour l'instant, vous vous posez peut-être les questions suivantes : pourquoi se donner la peine d'apprendre à programmer en BASIC puisque des programmes ont déjà été écrits par d'autres, et qu'il est possible de les acheter et de les utiliser ?

En fait, une grande partie des gens qui achètent un ordinateur ZX81 peuvent ne jamais apprendre à écrire un programme en BASIC. Ils choisiront et achèteront chez leur revendeur des programmes adaptés à leurs propres besoins. C'est une utilisation rationnelle de l'ordinateur; vous n'avez pas *besoin* d'apprendre à programmer pour utiliser avec profit votre ordinateur. Cependant, si vous n'avez jamais essayé d'apprendre le BASIC, et d'écrire vos propres programmes, vous aurez raté une bonne occasion d'auto-apprentissage. Si vous apprenez à concevoir vos propres programmes, votre ordinateur deviendra un outil de valeur que vous pourrez toujours utiliser pour vos besoins individuels en informatique. De plus, le processus d'apprentissage de la programmation d'un ordinateur vous donnera un nouvel aperçu sur un instrument qui prendra de plus en plus d'importance dans notre société.

RÉSUMÉ

Pour communiquer avec votre ordinateur, vous avez besoin de connecter des éléments d'entrée et de sortie d'informations. Sur le ZX81, l'organe principal d'introduction des informations est le clavier qui lui est intégré. Le clavier présente de façon commode toutes les commandes que vous pouvez donner à votre ordinateur, et chaque mot peut être introduit d'une seule pression de touche.

L'écran TV est la principale unité de sortie d'informations, et

elle fournit de nombreuses informations sur vos programmes, leurs résultats, et le fonctionnement de l'ordinateur à un instant donné. De plus vous pouvez connecter une imprimante à votre ordinateur si vous souhaitez conserver vos résultats de façon permanente.

Un simple magnétophone à cassettes peut également être relié à votre ZX81 quand vous voudrez sauvegarder les programmes que vous avez écrits, ou quand vous voudrez utiliser des programmes du commerce. Enfin, des modules de mémoire supplémentaire sont disponibles pour accroître la taille de la mémoire vive (RAM) de votre ordinateur.

Les mots que vous pouvez voir sur le clavier constituent le vocabulaire du langage BASIC. L'apprentissage de la programmation en BASIC n'est absolument pas nécessaire à l'utilisation d'un ordinateur, mais elle peut rendre votre expérience informatique plus efficace et plus intéressante.

Chapitre 2

LE PREMIER ACTE : INTRODUIRE VOTRE PROGRAMME

INTRODUCTION

Si vous avez suivi les instructions détaillées du Chapitre 1, votre ordinateur est sous tension, le cordon TV branché, et vous êtes prêt pour commencer à travailler. La première tâche qui vous attend est l'apprentissage de l'utilisation du clavier de l'ordinateur. C'est l'objectif essentiel de ce chapitre. Nous commencerons par jeter un bref et attentif regard au BASIC. Nous verrons à quoi ressemble un programme, et nous en présenterons la structure. Puis vous introduirez un court programme dans votre ordinateur. Parallèlement, vous verrez de quelle façon l'ordinateur vous aide à repérer et corriger toutes les erreurs que vous pouvez commettre.

Ensuite, vous verrez comment utiliser un magnétophone à cassettes avec votre ordinateur. Nous détaillerons soigneusement les branchements que vous devez effectuer et dans quel ordre, pour sauvegarder des programmes sur cassettes, et recharger un programme dans l'ordinateur depuis une cassette. Vous expérimenterez le processus en sauvegardant le programme que vous aurez saisi au clavier.

Ceci nous amènera au troisième thème : les programmes que vous pouvez acheter sur cassette. Nous étudierons un exemple et terminerons le chapitre par un guide d'achat des programmes pour votre ZX81.

L'ORDINATEUR APPREND SES LIGNES

Tout bon ordinateur doit naturellement apprendre à être amical; aussi le premier programme que nous examinerons guidera l'ordinateur dans les tâches suivantes :

1. vous demander votre nom,
2. tracer un dessin sur l'écran TV,
3. imprimer un salut sur l'écran.

Vous pouvez considérer ceci comme des travaux peu sérieux pour un programme d'ordinateur; mais en ce moment nous ne nous soucions pas d'utilité. Le but de ce programme est de vous fournir une introduction minutieuse à l'usage du clavier. Pendant que vous saisirez le programme, vous aurez des exemples de tout ce que le clavier peut faire pour vous.

Le langage d'ordinateur BASIC

Le programme SALUT est montré Figure 2.1. Prenez un moment pour l'étudier avant de commencer à le taper sur votre ordinateur. Vous pouvez apprendre un certain nombre de choses sur le langage BASIC, uniquement en examinant un programme BASIC.

Nous avons appris dans le Chapitre 1 que le BASIC a un vocabulaire très limité composé principalement de mots anglais usuels. En fait le programme SALUT ne comporte que peu de mots qui semblent être des abréviations (CLS, LEN, CHR$); la plupart des autres mots sont des mots anglais assez courants. Ceci ne signifie pas que vous pourrez immédiatement comprendre ce que chaque ligne du programme fait, mais au moins vous pourrez voir que le programme, en tous cas, ne semble pas être intimidant.

Que nous dit ce programme quant aux caractéristiques du BASIC ? D'abord un programme BASIC est organisé en lignes, et chaque ligne est numérotée. Ce programme comprend 13 lignes. La longueur d'un programme peut varier d'une seule à des centaines de lignes, en fonction de la tâche pour laquelle il est conçu.

Les lignes de ce programme sont numérotées de 10 à 110, mais remarquez que les intervalles ne sont pas réguliers. La plupart des numéros de lignes progressent de 10 en 10, mais quelques-uns de 5 en 5. Quand vous écrivez un programme BASIC, vous êtes libre de numéroter les lignes de la manière que vous voulez. Pour l'ordinateur, l'unique signification des numéros de lignes est d'indiquer l'ordre dans lequel les lignes doivent être traitées. Ce programme aurait pu être numéroté de 1 à 13 ou de 100 à 1300, et l'ordinateur l'aurait toujours accepté. L'incrémentation régulière

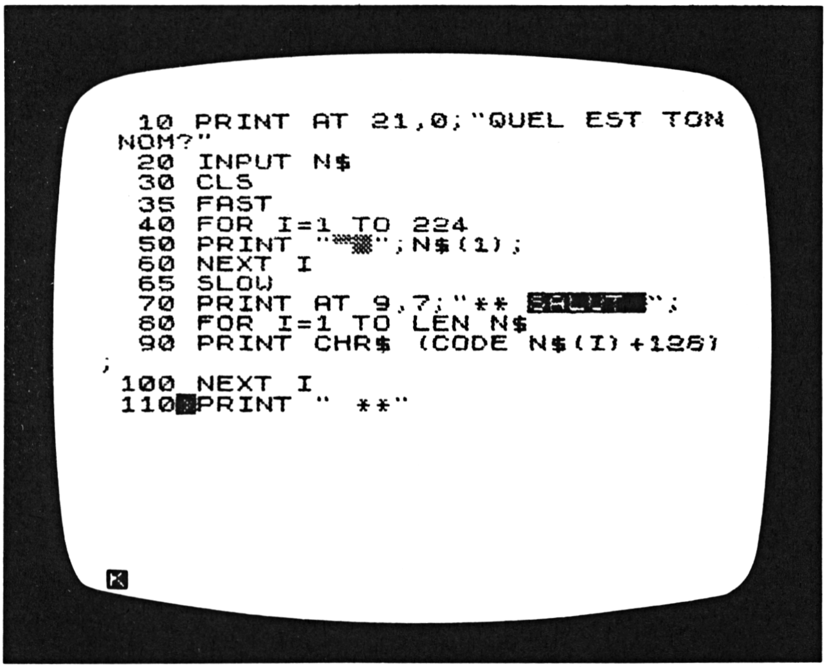


Figure 2.1 : Programme SALUT

des numéros de lignes, de 10 en 10 par exemple, offre un avantage : une fois le programme écrit, vous pourrez souhaiter lui ajouter des lignes, n'importe où à l'intérieur. Ceci sera facile tant que votre système de numérotation laissera de la place pour des lignes supplémentaires. Par exemple, les lignes 35 et 65 de ce programme ont été ajoutées après que le reste du programme ait été écrit.

Deux des lignes numérotées de ce programme, les lignes 10 et 90, occupent plus d'une ligne sur l'écran TV. C'est parfaitement correct. L'ordinateur organise l'information sur l'écran de façon que chaque *ligne d'écran* puisse occuper jusqu'à 32 caractères, mais une ligne de programme BASIC peut être beaucoup plus longue que cela. Quand une ligne du programme dépasse le nombre de caractères disponibles sur une ligne d'écran, elle est simplement continuée sur la ligne suivante. Ceci peut provoquer un certain nombre de coupes inattendues dans les lignes du programme – par exemple, le dernier caractère de la ligne 90, un point-virgule, a dû aller seul sur la ligne suivante, mais ceci n'affecte pas la façon dont l'ordinateur lit et exécute la ligne.

Le premier mot de chaque ligne d'un programme BASIC est toujours un *mot clé.* Les mots clés indiquent à l'ordinateur quelle sorte d'instruction se présente. Chaque mot clé doit être suivi de sa *grammaire* particulière, c'est-à-dire que l'ordinateur sait quelle sorte de modèle d'instruction attendre après lui. Votre clavier possède 26 mots clés qui se trouvent au-dessus de chacune des lettres. La Figure 2.2 montre la position des mots clés.

Lorsque vous tapez un numéro de ligne de programme BASIC sur votre ordinateur, celui-ci range la ligne en mémoire et l'affiche à l'écran. Il ne commencera à exécuter les instructions de votre programme que lorsque vous lui direz de le faire. Il conserve simplement la trace du programme, et attend que vous lui disiez quoi faire ensuite. Nous verrons comment cela se passe après avoir décrit la manière d'introduire le programme.

Le clavier : mots clés et modes

Actuellement votre écran est vide à l'exception du K en vidéo inversée, dans le coin gauche en bas de l'écran. Vous vous rappellerez que le K signifie *"mot clé"* (en Anglais : *Keyword*). Ce

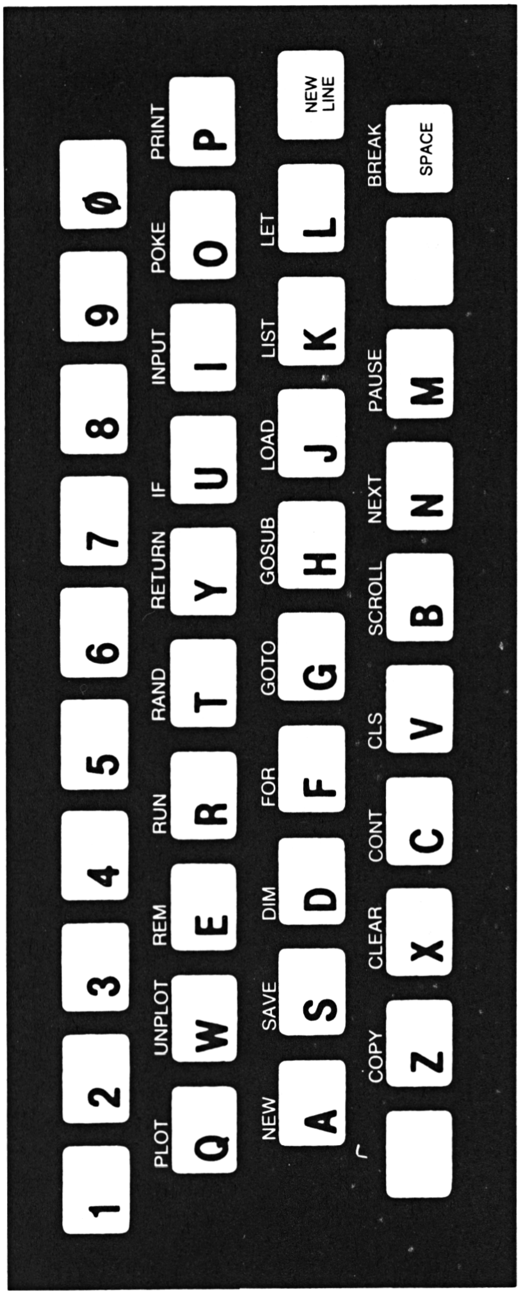


Figure 2.2 : Les mots clés

caractère indicateur (appelé *curseur* dans votre manuel-utilisateur) vous indique que l'ordinateur est prêt à accepter n'importe quel mot clé depuis le clavier. Quand vous commencerez à taper le programme, vous verrez d'autres caractères indicateurs – dont L, G, F et S, tous en vidéo inversée. Ces caractères indicateurs vous disent dans quel *mode* se trouve le clavier, c'est-à-dire quelle sorte de mots ou de symboles pourront être ensuite introduits dans l'ordinateur.

Nous examinerons chaque ligne du programme pour chaque pression de touche. Vous taperez les lignes une à une en lisant l'explication. (Vous lirez également rapidement la description générale de ce que fait chaque ligne, mais pour l'instant, concentrez-vous sur le clavier et non sur la signification des lignes. Vous aurez largement le temps ensuite, de maîtriser le BASIC une fois que vous aurez appris l'utilisation du clavier.)

Introduction du programme, ligne à ligne

La première ligne du programme est l'instruction PRINT. PRINT indique à l'ordinateur d'afficher une information sur l'écran TV. A la première ligne, PRINT est suivi du mot AT, qui est une des façons possibles de préciser l'endroit exact de l'écran, où vous voulez que l'information soit affichée. L'information ici est la question "QUEL EST TON NOM ?" apparaissant entre guillemets à la fin de l'instruction PRINT.

Tapons la ligne. Si vous avez déjà tapé quelque chose sur votre ordinateur, vous aurez remarqué que les touches ne demandent pas une forte pression pour être prises en compte. Une légère pression suffit. En tapant vous regarderez cependant l'écran pour être certain que chaque touche pressée a bien envoyé son *message* à l'ordinateur.

Comme nous l'avons vu, la première chose à taper au début de chacune des lignes de votre programme BASIC est le numéro de ligne. Pour la première ligne, entrez les chiffres 1 et 0. A chaque fois que vous introduirez un caractère, vous verrez le caractère indicateur K se déplacer d'une position vers la droite, et le chiffre apparaître à sa gauche.

D'abord, trouvez la commande PRINT. Elle se trouve au-dessus de la touche P (complètement à droite du clavier, à la seconde

rangée vers le bas). Appuyez sur la touche et voyez ce qui se passe sur l'écran. Le mot PRINT tout entier apparaît en une fois sur la ligne. De plus, le caractère indicateur inversé devient L. Ceci vous indique que le mode mot clé a été abandonné, et que vous êtes maintenant en mode lettre; la prochaine touche que vous enfoncerez sera interprétée en tant que lettre ou chiffre, et non en tant que mot clé.

Pour introduire un mot clé quelconque, la seule chose à faire est d'appuyer une seule fois sur la touche sur laquelle le mot clé est inscrit. En fait, vous ne pouvez pas frapper une à une les lettres d'un mot clé. Si vous tapiez les lettres P-R-I-N-T, l'ordinateur ne reconnaîtrait pas en cela le mot clé PRINT. Chaque fois que l'introduction d'un mot clé est correcte dans la syntaxe particulière au BASIC, l'ordinateur vous le dit en affichant le caractère indicateur K. Si vous ne voyez pas ce caractère indicateur K, en vidéo inversée, vous saurez alors que vous ne pouvez pas introduire de mot clé.

Le mot suivant de la première ligne est AT. AT est l'une des 25 *fonctions* disponibles sur votre clavier. Les fonctions apparaissent au-dessus des touches de lettres (hormis la touche V). Vous pouvez le voir Figure 2.3. Un certain nombre de ces fonctions sont de véritables fonctions mathématiques telles que le Sinus (SIN), Cosinus (COS), Logarithme Népérien (LN). Si vous avez l'intention d'utiliser votre ordinateur pour des travaux scientifiques ou mathématiques, alors c'est certain, vous connaissez déjà ces fonctions.

D'autres fonctions de votre clavier n'ont rien de mathématique. Vous pouvez les considérer comme des fonctions de programmation, car une fois que vous saurez vous en servir, elles simplifieront de nombreuses tâches de programmation. AT, par exemple, est une *fonction* vous permettant de préciser très facilement l'endroit où l'ordinateur doit placer l'information sur l'écran.

Toutes les fonctions, mathématiques ou autres, ont quelque chose en commun. Pour taper une fonction vous devez faire *passer* le clavier en mode fonction. Pour ce faire, appuyez sur 2 touches à la fois :

- la touche SHIFT, qui se trouve dans le coin gauche, en bas du clavier, et

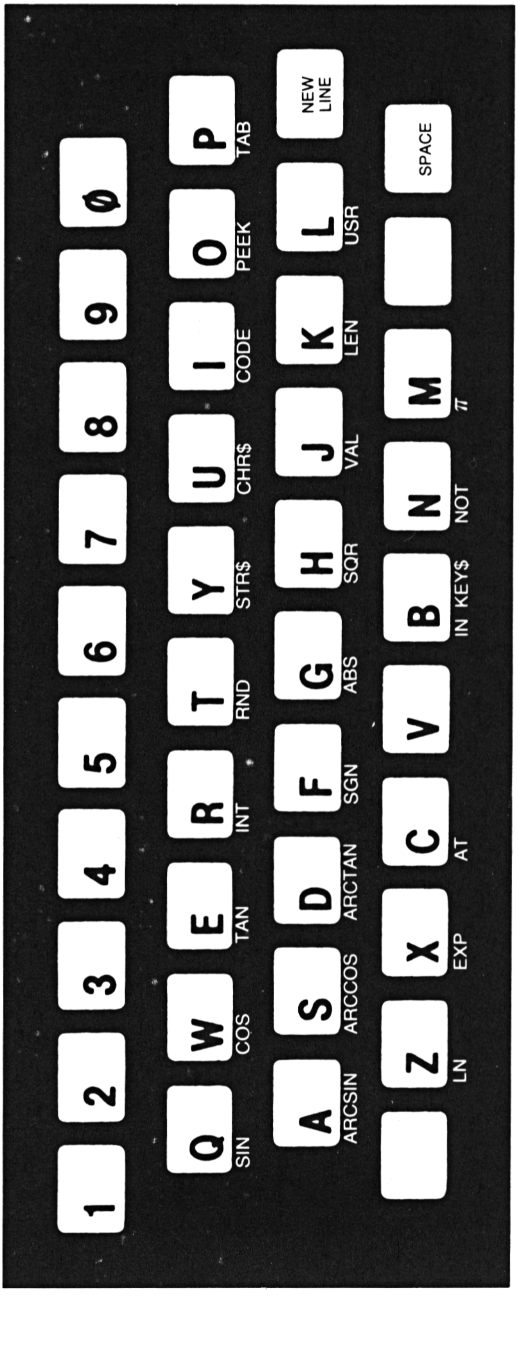


Figure 2.3 : Les fonctions

- la touche FUNCTION/NEW LINE (ou FUNCTION/ENTER), qui se trouve à droite du clavier, juste en dessous de la touche P.

Tenez d'un doigt la touche SHIFT enfoncée, puis appuyez sur la touche FUNCTION, et voyez ce qui se passe sur l'écran. Le caractère indicateur en vidéo inversée se transforme de L en F. Vous êtes maintenant en mode fonction. Vous pouvez appuyer sur une quelconque des touches lettres ayant une fonction au-dessous d'elles, et la fonction apparaîtra sur l'écran comme terme de votre ligne BASIC.

Repérer la fonction AT sur votre clavier. Elle est au-dessous de la touche C. Appuyez sur la touche, et AT s'intégrera à votre ligne.

On lit maintenant la ligne :

```
10 PRINT AT
```

Remarquez que le caractère indicateur en vidéo inversée s'est à nouveau transformé en L.

De même que les mots clés, les fonctions doivent être introduites d'une seule pression de touche. Taper les lettres A et T en mode lettre ne signifie pas la même chose pour l'ordinateur que la fonction AT, en mode fonction.

Ensuite, nous avons deux nombres à taper sur la ligne. Comme vous l'avez peut-être deviné, ces nombres représentent la *position* de l'écran à laquelle l'ordinateur imprimera le message. (Les nombres 21,0 signifient ligne 21 et colonne 0; nous verrons exactement comment cela se passe, en étudiant la fonction AT, au Chapitre 3.)

Avant d'aller plus loin dans l'étude du clavier, voyons ce que nous devons faire en cas d'erreur de frappe. De telles erreurs sont inévitables; peut-être en avez-vous déjà fait une, et êtes-vous quelque peu embarrassé pour continuer ?

L'ordinateur vous offre une méthode très simple, pour *effacer* quoi que ce soit, ayant été introduit dans une ligne. A nouveau, cela nécessite d'appuyer sur deux touches à la fois :

- la touche SHIFT, et
- la touche 0 (Zéro).

Vous remarquerez deux choses relatives à la touche 0 (Zéro) située en haut à droite du clavier. D'abord, pour distinguer le caractère 0 (Zéro) de la lettre O, le zéro possède un trait oblique.

(Ce trait oblique apparaît également à l'écran.) Ensuite, la touche Zéro possède le mot RUBOUT (ou DELETE, selon les modèles ZX81 ou Timex Sinclair 1000), écrit en rouge au-dessus du chiffre. Ainsi, en appuyant sur les touches SHIFT et RUBOUT (ou DELETE) en même temps, vous pouvez effacer les fautes éventuellement commises.

Maintenant essayez une fois. Entrez quelques caractères ne devant pas se trouver sur votre ligne.

Puis tenez la touche SHIFT enfoncée avec un doigt, et appuyez sur la touche RUBOUT (ou DELETE) avec un autre. Remarquez bien ce qui se passe sur l'écran. Le caractère indicateur L saute en arrière par-dessus le caractère indésirable qui disparaît, et vous pouvez continuer votre saisie.

Entrez maintenant 2 nombres : d'abord 21 suivi d'une virgule (,), puis 0 suivi d'un point-virgule (;). Cette ponctuation n'ayant pour vous peut-être aucune signification immédiate, est cependant essentielle pour le sens de la ligne, aussi doit-elle être introduite correctement. La virgule se trouve dans la zone de droite, vers le bas du clavier, sur la même touche que le point. Le point-virgule est situé sur la touche X. Virgule et point-virgule apparaissent tous deux en rouge sur le clavier.

En fait, chaque touche du clavier possède un caractère ou un mot imprimé en rouge. Nous les désignerons comme caractères *majuscules* (SHIFT). Pour les taper vous devez d'abord tenir la touche SHIFT enfoncée, puis avec un autre doigt, presser la touche appropriée. Tous les caractères majuscules apparaissent à la Figure. 2.4.

Essayez de taper la virgule. Si vous obtenez d'abord un point au lieu d'une virgule, c'est que vous ne tenez pas tout à fait bien enfoncée la touche SHIFT, ou que vous n'avez pas appuyé assez tôt sur cette dernière avant d'appuyer sur la touche VIRGULE. Supprimez le point (SHIFT-RUBOUT ou SHIFT-DELETE) et essayez à nouveau. Vous devez maintenant avoir sur l'écran la ligne :

```
10 PRINT AT 21,0;
```

Le reste de la ligne est le message à imprimer sur l'écran par l'ordinateur lorsqu'il exécutera l'instruction PRINT. Le message est entre guillemets. Le guillemet se trouve sur la touche P, en tant

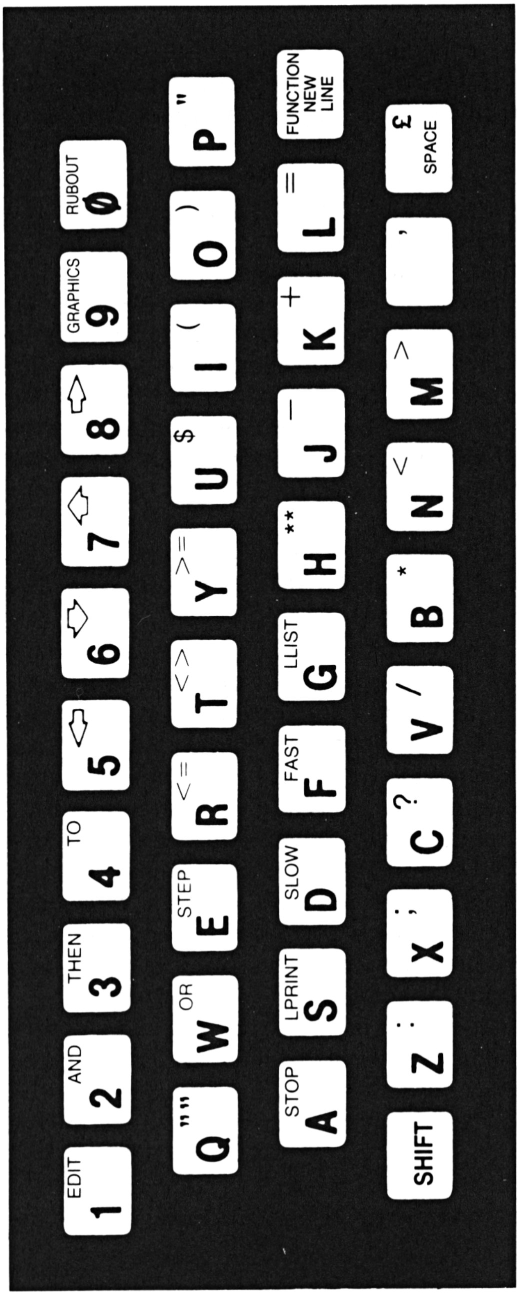


Figure 2.4 : Les caractères majuscules

que caractère majuscule. Faites SHIFT-P; le guillemet doit s'inscrire à l'écran.

Tapons maintenant le message : QUEL EST TON NOM ?, caractère par caractère (la touche d'espacement se trouve dans le coin en bas à droit du clavier : SPACE). Vous remarquerez qu'en tapant le N de NOM, vous êtes sorti de la place disponible sur la ligne d'écran. En tapant O, l'ordinateur saute automatiquement à la ligne suivante. Le message se termine par un point d'interrogation, caractère majuscule situé sur la touche C.

Avant de terminer par le guillemet, nous allons expérimenter une autre caractéristique spéciale de votre ordinateur. Normalement, vous devriez terminer la saisie de la ligne, puis appuyer sur la touche NEW LINE (ou ENTER) pour dire à l'ordinateur que la ligne est complète. Voyons maintenant ce qui se passe si vous avez fait une faute : appuyez sur NEW LINE (ou ENTER) *avant* de taper le guillemet final.

Détection automatique des erreurs

Ce que vous voyez sur l'écran, après avoir appuyé sur la touche NEW LINE (ou ENTER) apparaît Figure 2.5.

Un nouveau caractère indicateur suit le L. Le caractère indicateur S représente une *erreur de syntaxe.* Le mot *syntaxe* fait référence aux règles de la *grammaire* particulière que vous devez appliquer en écrivant un programme BASIC.

Quand vous écrivez une ligne qui transgresse l'une de ces règles grammaticales, l'ordinateur l'indique à droite, et refuse de la prendre en compte en tant que partie de programme. Le caractère indicateur S est la façon polie de l'ordinateur de vous dire que vous vous êtes trompé. Vous devez regarder à nouveau la ligne en question.

Naturellement, puisque vous avez commis cette faute volontairement, vous savez exactement ce qui ne va pas. Le guillemet final manque. Maintenant, en appuyant sur GUILLEMET (SHIFT-P) le caractère indicateur S disparaît, et notre ligne est complète. Appuyez sur la touche NEW LINE (ou ENTER) à nouveau, et regardez ce qui arrive. La Figure 2.6 montre le résultat. La ligne de programme correcte s'est déplacée en haut de l'écran. Cela veut

dire que l'ordinateur l'a rangée en mémoire en tant que première ligne du programme (et en ce moment la seule). Le caractère indicateur K est réapparu dans le coin en bas à gauche de l'écran, ce qui vous indique que l'ordinateur est, une fois de plus, prêt pour de nouvelles commandes.

La deuxième ligne : utilisation de l'éditeur

Maintenant, introduisons au clavier la deuxième ligne du programme, la ligne 20 :

```
20 INPUT N$
```

Le mot clé INPUT indique à l'ordinateur d'interrompre le déroulement du programme, temporairement, et d'attendre que

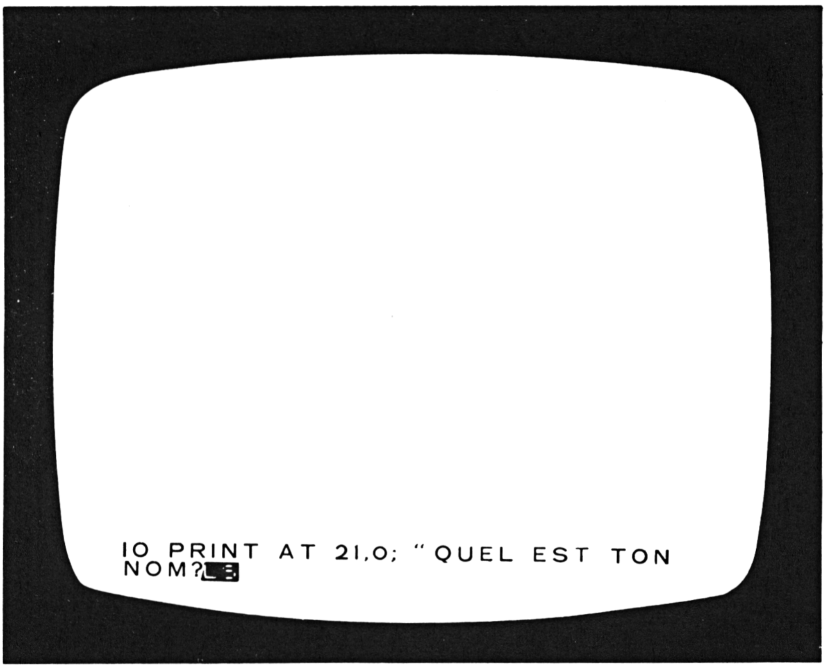


Figure 2.5 : L'ordinateur répond à une erreur de syntaxe

vous ayez introduit une information au clavier. En utilisant l'instruction INPUT dans votre programme, vous créez une sorte de *dialogue* entre l'ordinateur et son utilisateur. Vous verrez exactement comment fonctionne ce processus, un peu plus loin, une fois que vous aurez terminé la saisie du programme, et que vous le ferez fonctionner.

Les caractères N$ qui suivent le mot INPUT représentent ce que nous appelons un *nom de variable.* Que cette terminologie ne vous intimide pas ! Une variable est tout simplement un emplacement que l'ordinateur réserve dans sa mémoire pour une information de nature précise. Vous pouvez définir de nombreuses variables dans le but de les utiliser dans un programme, mais il faut donner un nom à chacune d'entre elles; l'ordinateur peut retrouver les variables grâce à leurs noms. L'information stockée dans une variable peut changer à maintes reprises

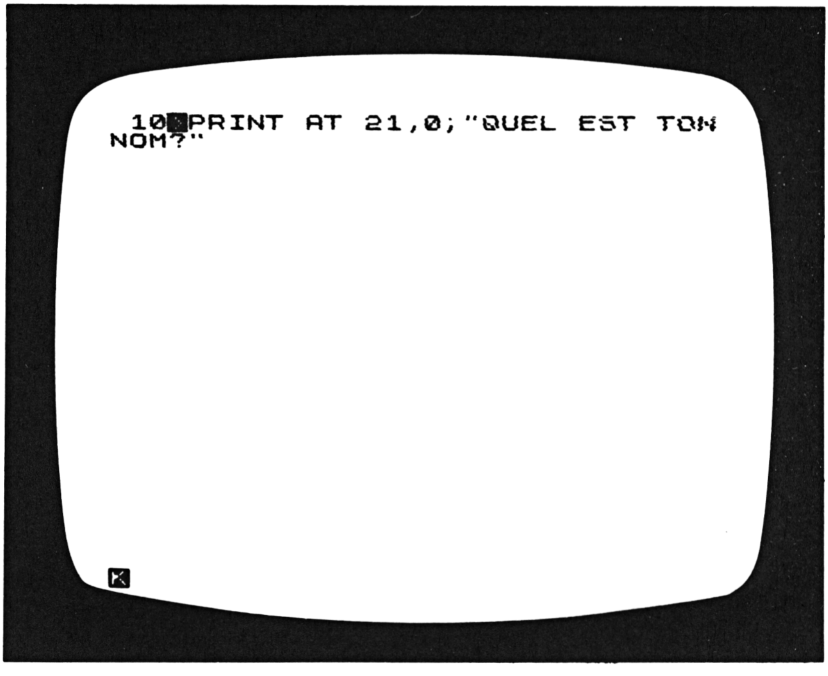


Figure 2.6 : Entrer une ligne de programme correctement

pendant le déroulement du programme – c'est pourquoi on utilise le mot "variable" – mais le *nom* n'est jamais modifié.

Quand l'ordinateur exécute l'instruction INPUT N$, il commence par attendre qu'une information soit introduite depuis le clavier. Dès que cela est fait, l'ordinateur range cette information dans une variable qu'il nomme N$. Désormais, chaque fois que vous ferez référence au nom N$, l'ordinateur saura que vous êtes intéressé par l'information stockée dans cette variable.

Vous pouvez définir deux types de variables utilisables à l'intérieur d'un programme – des variables destinées à stocker de l'information numérique, et des variables prévues pour du texte (de l'information non numérique). Dans la terminologie informatique, nous appelons ce type d'information non-numérique, une *chaîne de caractères.* Pour définir une variable chaîne de caractères, vous devez indiquer, *dans le nom de la variable,* que vous voulez que l'ordinateur réserve en mémoire de la place pour une chaîne de caractères. Vous le faites en ajoutant le symbole "dollar" ($) à la fin du nom de la variable.

Ainsi, disons-le une fois de plus, INPUT N$ demande à l'ordinateur de lire une chaîne de caractères depuis le clavier et de la ranger en mémoire sous le nom N$. Si, à ce stade, le concept de variable vous paraît encore un peu confus, ne vous tracassez pas trop. Nous y reviendrons plus tard. Souvenez-vous que notre objectif primordial est actuellement d'apprendre à nous servir du clavier. Si nous abordons dès maintenant le sujet des noms de variables, c'est pour vous aider à comprendre une nouvelle fonction de correction des erreurs que possède votre ordinateur : l'éditeur.

En introduisant la ligne 20 dans votre ordinateur, commencez par le faire de la façon suivante, en omettant volontairement le symbole "$" :

```
20 INPUT N
```

Du point de vue de la syntaxe, c'est une instruction tout à fait correcte. Elle signifie : "Lire une information *numérique* depuis le clavier, et ranger cette information en mémoire, sous le nom N". Puisque le symbole "$" indiquant une variable chaîne de caractères manque après "N", l'ordinateur interprète l'instruction comme étant relative à une variable *numérique.* Pour l'instant,

introduisez ainsi cette ligne. Le mot clé INPUT, se trouve au-dessus de la touche I. Introduisez la ligne au clavier, puis appuyez sur la touche NEW LINE (ou ENTER). La ligne sautera en haut de l'écran, juste en dessous de la ligne 10.

Souvent, en écrivant un programme, vous pourrez commettre ce genre de faute. Vous avez introduit une ligne correcte de programme, mais ce n'était pas exactement la ligne que vous vouliez. Puisque la ligne est correcte syntaxiquement l'ordinateur n'a aucun moyen de savoir que quelque chose ne va pas, et accepte la ligne telle qu'elle a été introduite. Une fois la ligne introduite, et en examinant le programme à l'écran, vous reconnaîtrez probablement l'erreur, et voudrez la corriger. L'ordinateur ZX81 fournit un moyen extrêmement simple de correction de telles erreurs. Ce moyen s'appelle *l'éditeur.*

Comme la ligne que vous souhaitez *éditer* est celle que vous venez d'introduire, vous pouvez utiliser directement l'éditeur.

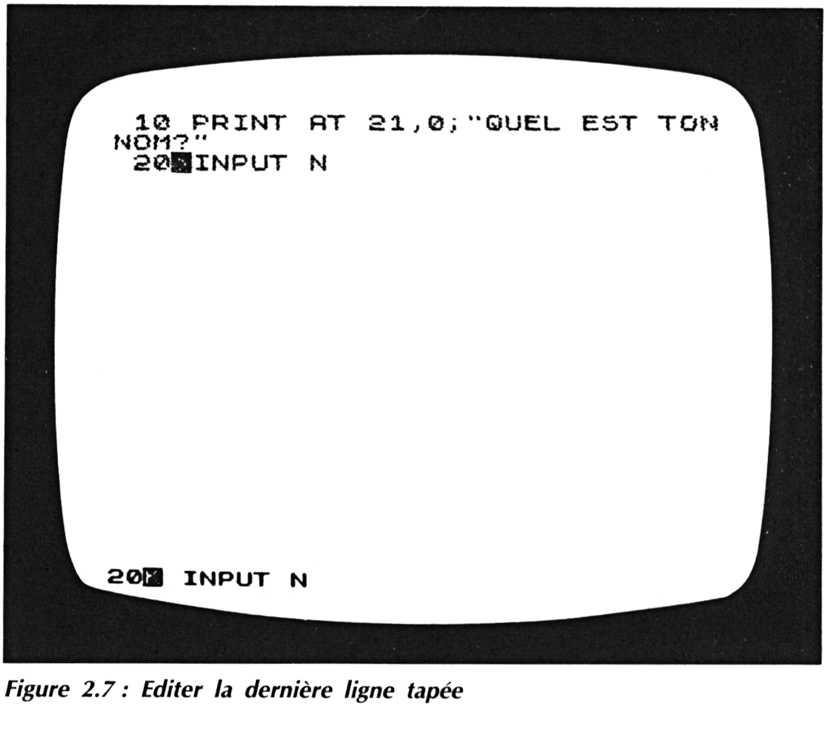


Figure 2.7 : Editer la dernière ligne tapée

Trouvez le mot EDIT sur votre clavier : il se trouve sur la touche "1". EDIT est écrit en rouge, donc vous savez qu'il nécessite un SHIFT. Tenez la touche SHIFT enfoncée, puis appuyez sur la touche EDIT. La Figure 2.7 montre ce qui va se passer à l'écran.

Vous pouvez voir qu'une copie de la ligne a été redescendue depuis le haut de l'écran, dans la zone où apparaissent les nouvelles lignes que vous entrez. Le caractère indicateur K apparaît à l'intérieur de la ligne, juste après le numéro de ligne. Vous pouvez maintenant utiliser les touches *Flèche gauche* et *Flèche droite* (respectivement SHIFT-5 et SHIFT-8) pour déplacer le curseur vers l'arrière ou vers l'avant dans la ligne. Pour les suppressions, vous pouvez déplacer le curseur juste à droite du caractère ou du mot que vous voulez supprimer, et appuyer sur SHIFT-RUBOUT (ou SHIFT-DELETE) exactement comme si c'était une nouvelle ligne.

Dans notre cas, vous voulez ajouter un caractère à la ligne, plus précisément le caractère $ à la fin du nom de variable. Appuyez sur SHIFT-8 (c'est-à-dire en maintenant SHIFT enfoncée, appuyez sur la touche "8") deux fois pour déplacer le curseur à la bonne position, juste après le "N". Le caractère $ est aussi un caractère majuscule, et se trouve au-dessus de la touche "U". Appuyez sur SHIFT-U pour ajouter le caractère à la ligne. Puis appuyez sur NEW LINE (ou ENTER) pour ré-introduire la version *éditée* de la ligne. Vous verrez la ligne corrigée apparaître en haut de l'écran. Et, une fois de plus, le caractère indicateur K apparaîtra dans le coin gauche, en bas; l'ordinateur est prêt pour la commande suivante.

Vous n'avez pour l'instant introduit que deux lignes de programme, mais en le faisant vous avez appris pas mal de choses sur votre clavier et votre ordinateur. Résumons brièvement ce que vous avez appris :

- Quand vous commencez à introduire une ligne dans votre ordinateur, elle apparaît en bas de l'écran. Si vous faites, à ce moment, une erreur de frappe, vous pouvez "supprimer" la dernière pression de touche en appuyant en même temps sur les touches SHIFT et RUBOUT (ou DELETE).
- Le premier mot de chaque ligne est un mot clé que vous introduisez d'une seule pression de touche. Ensuite, le

caractère indicateur se transforme en L en vidéo inversée, pour le *mode lettre.* Si vous souhaitez introduire une fonction, vous devez d'abord passer en *mode fonction* en appuyant sur SHIFT-FUNCTION. Toutes les fonctions sont également introduites d'une seule pression de touche.

- Tous les caractères et mots qui apparaissent en rouge sur votre clavier sont des caractères majuscules. Pour introduire l'un d'entre eux, vous devez d'abord tenir la touche SHIFT enfoncée puis seulement appuyer sur la touche appropriée.
- Lorsque vous avez terminé une ligne, vous appuyez sur la touche NEW LINE (ou ENTER) pour dire à l'ordinateur de ranger la ligne, composante du programme. Si la ligne ne contient aucune erreur de syntaxe, elle est déplacée en haut de l'écran, avec le reste de la liste du programme.
- Si vous essayez d'introduire une ligne comportant une erreur de syntaxe, l'ordinateur détectera l'erreur et affichera le caractère indicateur S pour vous signaler où se trouve l'erreur. Vous pouvez utiliser les flèches gauche et droite (SHIFT-5, SHIFT-8) pour déplacer le curseur à l'intérieur de la ligne et la touche RUBOUT (ou DELETE) pour supprimer une erreur quelconque. Vous pouvez également ajouter des mots ou des caractères, si nécessaire.
- Pour corriger une erreur dans une ligne déjà introduite dans le programme, utilisez comme méthode EDIT, que l'on appelle en entrant SHIFT-1.

Nous allons passer plus rapidement sur les autres lignes du programme, mais gardez en mémoire les facilités de correction d'erreurs de l'ordinateur, pour les utiliser si besoin est. Quand vous aurez terminé, votre écran ressemblera à la liste du programme de la Figure 2.1.

Fin du programme

Les deux lignes suivantes, 30 et 35, sont constituées toutes deux d'une instruction en un mot. La ligne 30 est :

```
30 CLS
```

Le mot clé CLS, qui se trouve au-dessus de la touche "V", commande à l'ordinateur d'effacer toutes les informations se trouvant sur l'écran. L'instruction de la ligne 35 met l'ordinateur dans le mode *calcul rapide* dont nous parlerons dans les Chapitres 3 et 4 :

```
35 FAST
```

Le mot FAST est un caractère majuscule situé sur la touche "F".

Les 3 lignes suivantes (40 à 60) agissent ensemble pour indiquer à l'ordinateur de répéter plusieurs fois la même tâche. Nous appelons l'opération qui résulte de cette série de lignes une *boucle,* car l'ordinateur exécute les lignes, puis revient en arrière pour les exécuter à nouveau, et ainsi de suite. Les mots clés qui définissent cette sorte de boucle sont FOR (ligne 40) et NEXT (ligne 60). Dans cet exemple particulier, la tâche à répéter consiste en une seule ligne (la ligne 50), mais on peut construire une boucle qui répète de nombreuses lignes. La fin d'une boucle est toujours constituée d'une ligne NEXT. La ligne FOR, outre qu'elle précise le point de départ de la boucle, indique exactement le nombre de fois que la tâche doit être répétée. Nous examinerons en détail la syntaxe de la boucle FOR-NEXT dans le Chapitre 3.

En voyant les résultats de ce programme, il ne devrait pas vous être difficile de deviner en quoi consiste la tâche répétée. Introduisez la ligne 40 :

```
40 FOR I = 1 TO 224
```

Le mot clé FOR est situé au-dessus de la touche "F". Le signe égal (=) et le mot TO sont tous deux des caractères majuscules, se trouvant respectivement sur les touches "L" et "4".

La ligne 50 introduit une nouvelle caractéristique du clavier. Revenez à la Figure 2.1 et regardez à la ligne 50 en quoi consiste cette nouvelle caractéristique. La ligne est faite d'une instruction PRINT. Après le mot clé, il y a deux *caractères graphiques* entre guillemets. Le ZX81 possède 20 caractères graphiques, pouvant être affichés dans n'importe quel ordre, en une position quelconque sur l'écran TV. Sur le clavier ces caractères sont situés sur les huit premières touches de chiffres, et sur douze des touches de lettres, comme on le voit à la Figure 2.8.

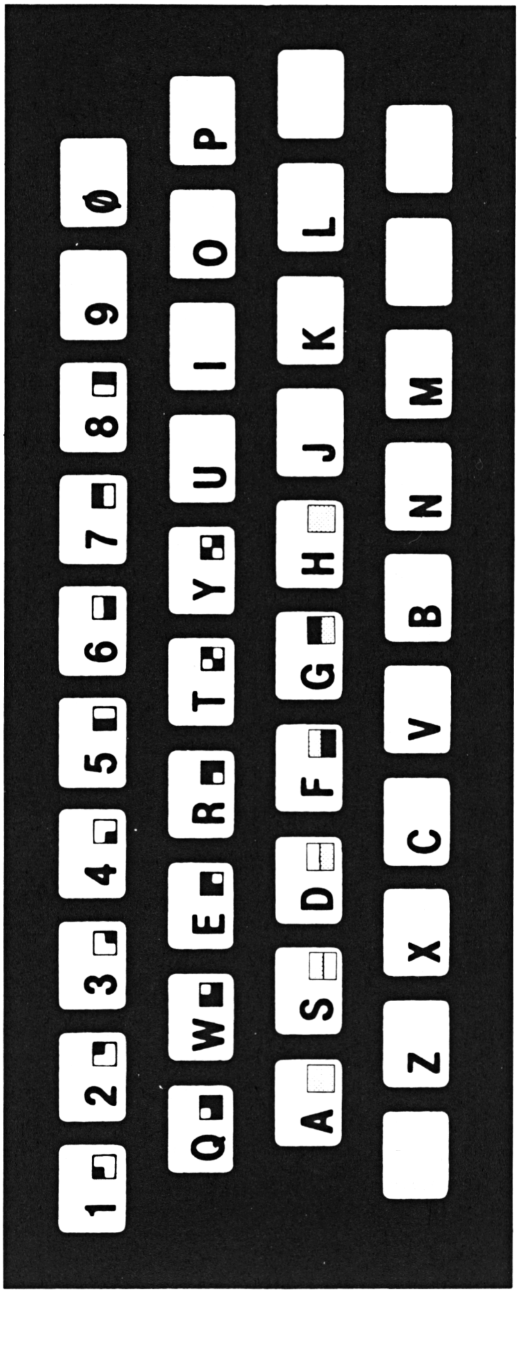


Figure 2.8 : Les caractères graphiques

Pour introduire ces caractères, vous devez d'abord passer en *mode graphique.* Commencez la ligne en introduisant le numéro de ligne, le mot clé, et le guillemet d'ouverture :

```
50 PRINT "
```

Le caractère indicateur "L" vous indique que vous vous trouvez actuellement dans le mode lettre. Pour passer en mode graphique, appuyer sur la touche SHIFT, puis la touche "9" avec un autre doigt. (Remarquez que la touche "9" possède le mot GRAPHICS, en rouge, au-dessus du "9".) En faisant ceci, le curseur va se changer en G en vidéo inversée, ce qui vous indique que vous êtes maintenant en mode graphique.

Tous les caractères graphiques de votre clavier sont des caractères majuscules. Les deux caractères dont vous avez besoin pour cette ligne se trouvent sur les touches "S" et "H". Commencez par appuyer sur SHIFT-S. Le caractère apparaîtra sur la ligne, l'ordinateur restant en mode graphique. Maintenant, appuyez sur SHIFT-H pour le second caractère. La Figure 2.9 montre ce que votre écran doit comporter à ce stade.

Pour terminer la ligne, il vous reste à quitter le mode graphique pour revenir en mode lettre. Pour ce faire, appuyez tout simplement à nouveau sur SHIFT-9. Souvenez-vous seulement que le mot GRAPHICS, situé sur la touche "9", vous permet d'entrer et de sortir du mode graphique.

Revenu en mode L, vous pouvez terminer la saisie de la ligne en introduisant la suite de caractères suivante :

```
";N$(1);
```

Toute la ponctuation est importante, aussi n'oubliez rien. Le 1 entre parenthèses, après N$, fait référence au *premier caractère* de la chaîne de caractères stockée en tant que variable N$ (vous vous en souviendrez en voyant les résultats du programme). Les parenthèses ouvrante et fermante sont toutes deux des caractères majuscules, situés respectivement sur les touches "I" et "O".

La ligne 60 termine la boucle :

```
60 NEXT I
```

Le mot clé NEXT se trouve au-dessus de la touche "N". La ligne 65 ramène l'ordinateur en mode calcul lent dont nous parlerons dans les Chapitres 3 et 4 :

65 SLOW

Le mot SLOW est un caractère majuscule, situé sur la touche "D".

La ligne 70 utilise une fois de plus les possibilités graphiques particulières du ZX81. Le mode graphique vous permet d'introduire des caractères en vidéo inversée, en plus des 20 caractères graphiques spéciaux. Pour voir comment cela fonctionne, commencez par introduire la première partie de la ligne :

70 PRINT AT 9,7;" **

(Après le guillemet ouvrant, il y a un espace, puis deux astérisques. L'astérisque est un caractère majuscule qui se trouve

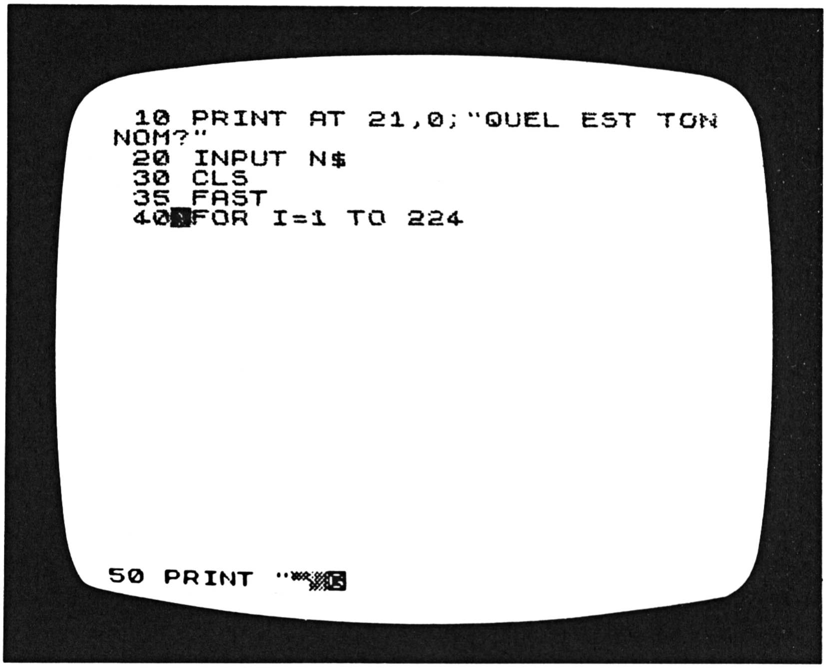


Figure 2.9 : Taper les caractères graphiques

sur la touche B.) Puis passez en mode graphique (SHIFT-9), et tapez le mot SALUT, sans appuyer sur la touche SHIFT. Vous verrez que le mot tout entier apparaîtra en caractères blancs sur fond noir – ce qui s'appelle la vidéo inversée. Puis appuyez sur la touche d'espacement; un espace en vidéo inversée consiste simplement en un carré noir. L'espace terminant le message, sortez du mode graphique, et tapez les deux derniers caractères de la ligne – le guillemet final, et le point-virgule. Appuyez sur NEW LINE (ou ENTER) pour ajouter la ligne au programme.

Les trois dernières lignes du programme – 80 à 100 – constituent une autre boucle FOR-NEXT. Mais maintenant, vous ne devriez pas rencontrer de difficultés pour introduire ces lignes. Elles mettent en œuvre trois nouvelles fonctions de programmation, si bien que vous devrez passer trois fois en mode fonction. La ligne 80 utilise la fonction LEN qui se trouve au-dessous de la touche K :

```
80 FOR I = 1 TO LEN N$
```

La fonction LEN fournit la *longueur* d'une chaîne de caractères, en nombre de caractères. La ligne 90 utilise la fonction CHR$ (au-dessous de la touche U) et la fonction CODE (au-dessous de la touche I) :

```
90 PRINT CHR$(CODE N$(I)+128);
```

Ceci est peut-être l'instruction la plus complexe du programme. En peu de mots, elle convertit les caractères de la chaîne N$ en leurs équivalents en vidéo inversée. Nous étudierons les fonctions CHR$ et CODE au Chapitre 5. La ligne 90 comporte également un nouvel élément : le signe Plus (+). C'est un caractère majuscule situé sur la touche K. La ligne 100 termine la boucle FOR-NEXT :

```
100 NEXT I
```

Et enfin, la ligne 110 est une autre instruction PRINT :

```
110 PRINT " **"
```

Remarquez qu'il y a trois caractères entre les guillemets : d'abord un espace, puis deux astérisques.

Vous avez maintenant introduit entièrement le programme dans

l'ordinateur, et vous êtes prêt à voir ce qui va se passer quand ce dernier exécutera le programme. Mais avant, faites deux choses. D'abord comparez soigneusement l'écran, ligne à ligne, avec la liste du programme présentée Figure 2.1. Assurez-vous que chaque caractère est absolument correct. Si vous trouviez des erreurs, ce ne serait pas difficile de les corriger. En sautant plus avant dans ce chapitre au paragraphe intitulé "Edition d'une ligne quelconque du programme", vous verrez comment effectuer des modifications supplémentaires à votre programme. (Vous savez déjà comment utiliser l'éditeur; tout ce que vous avez à apprendre est la façon de spécifier quelle ligne vous souhaitez éditer.)

La seconde chose à faire, avant d'exécuter le programme, est de passer soigneusement en revue tout ce que vous avez appris sur les modes du clavier. Etudiez la Figure 2.10, qui résume à votre intention les différents caractères indicateurs.

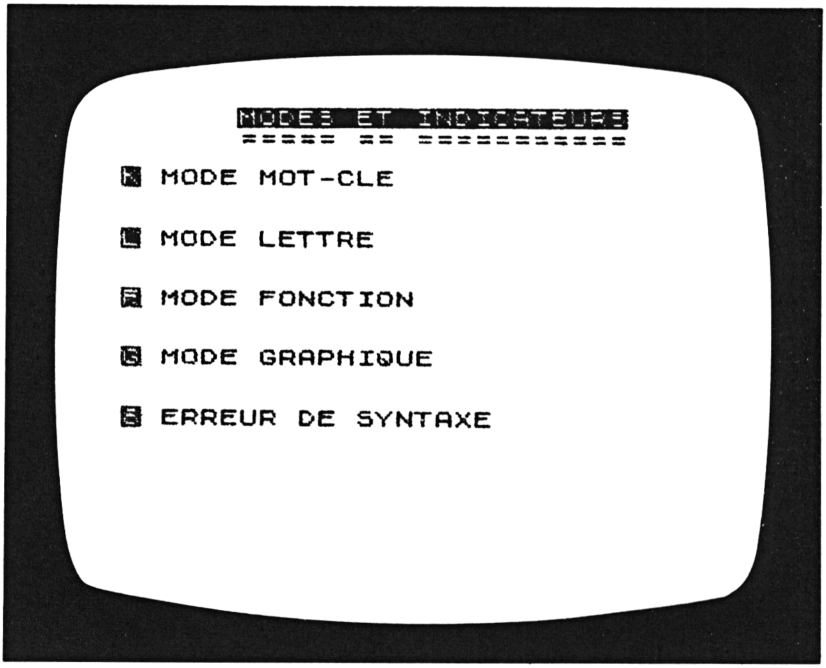


Figure 2.10 : Modes et indicateurs

L'EXÉCUTION

Le mot clé RUN, situé au-dessus de la touche R, indique à l'ordinateur de commencer l'exécution des instructions du programme qui est en mémoire. A l'exécution de ce programme à la saisie duquel vous avez tant travaillé, vous reconnaîtrez probablement l'origine d'au moins une partie des actions des lignes les plus compréhensibles, parmi celles que vous avez introduites. A ce stade, puisque vous n'avez pas encore étudié le BASIC, votre idée de la façon dont travaille le programme sera plus intuitive que logique. Mais vous serez à nouveau surpris de la précision avec laquelle vous devinerez le sens de la plupart des lignes du programme, une fois que vous aurez vu ce qu'elles font.

Votre dialogue avec l'ordinateur

Appuyez directement sur le mot clé RUN sans taper de numéro avant. Appuyez sur la touche NEW LINE (ou ENTER), et regardez ce qui se produit.

D'abord, la liste du programme a disparu de l'écran. Naturellement, les lignes sont toujours stockées en mémoire dans l'ordinateur, mais maintenant l'écran est réservé à l'affichage des résultats du programme.

L'exécution se passe en deux étapes distinctes, comme nous allons bientôt le voir. En premier lieu, l'interrogation :

```
QUEL EST TON NOM ?
```

apparaît à l'écran. Le caractère indicateur L, entre guillemets, est affiché en dessous de la question. Ceci vous indique que l'ordinateur attend que vous introduisiez une réponse au clavier. Allez-y, introduisez votre nom. (La Figure 2.11 montre ce que cela donnera avec le nom BRIGITTE.) Si vous vous trompez en tapant, vous pouvez utilisez la touche RUBOUT (ou DELETE), SHIFT-0, pour effacer l'erreur, exactement comme vous l'aviez fait lors de l'introduction du programme.

Une fois que vous aurez tapé votre nom, appuyez sur la touche NEW LINE (ou ENTER). L'écran reste blanc pendant quelques

instants, puis... vous verrez bien. (La Figure 2.12 montre le "salut Brigitte".) Remarquez la façon habile avec laquelle l'ordinateur a incorporé l'initiale de votre nom au dessin du fond de l'écran.

Avant de continuer, vous devez vous arrêter pour réfléchir un moment à l'illusion créée par votre programme. Cela se passe comme si l'ordinateur vous avait posé une question parfaitement polie, puis avait pris en compte votre réponse, et enfin produit une réponse, quelque peu extravagante, mais correcte, et semblant intelligente. Naturellement, vous le savez maintenant, ce "dialogue" est une sorte de trucage dans la mesure où chaque action est entièrement déterminée par le programme que vous avez introduit dans l'ordinateur. L'ordinateur ne se soucie en rien de votre nom, et rien ne le prédispose à être amical. Mais vous pouvez créer l'*illusion* de dispositions amicales – ou de n'importe quel autre genre de réaction – par la façon dont vous écrivez vos programmes. De plus le fait qu'il existe un programme quelque

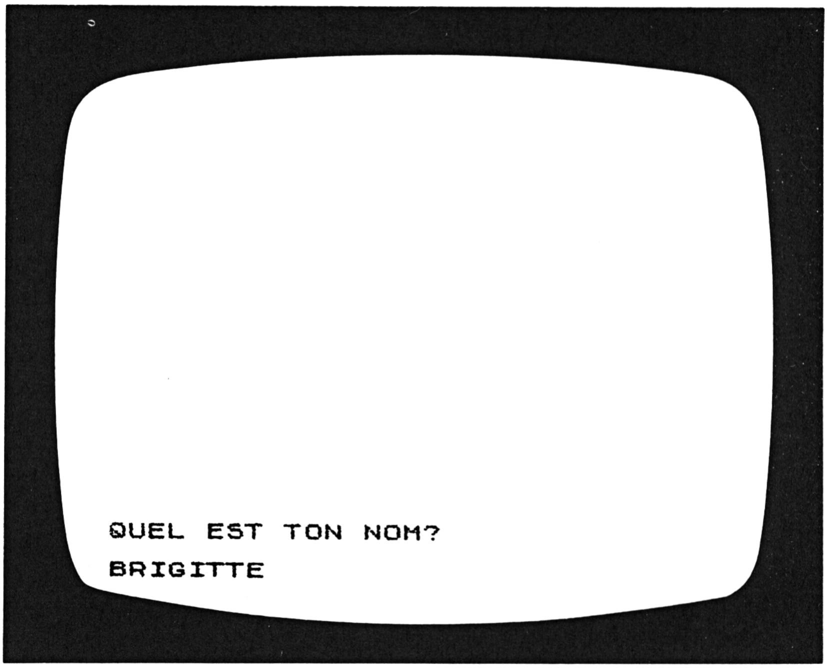


Figure 2.11 : Réponse à une question sur l'écran

part est totalement invisible à la personne qui répond à la question affichée à l'écran (à titre d'expérience, exécutez à nouveau le programme pour quelqu'un d'autre, et voyez comment il réagira). C'est un domaine essentiel de la programmation des ordinateurs que de créer l'illusion d'une intelligence, en fournissant des réponses correctes à ce que l'homme a introduit. L'ordinateur ne peut rien faire sans des instructions précises pour le guider, mais si par facétie vous apprenez à écrire des programmes habiles, cette habileté sera attribuée à l'ordinateur. Personne ne saura que c'est le programme qui fait tout.

Revenons à l'écran TV, et remarquez qu'un message bref apparaît dans le coin en bas à gauche. L'ordinateur a maintenant terminé l'exécution de toutes les lignes du programme et les nombres du coin représentent une sorte de compte rendu des résultats du programme :

0/110

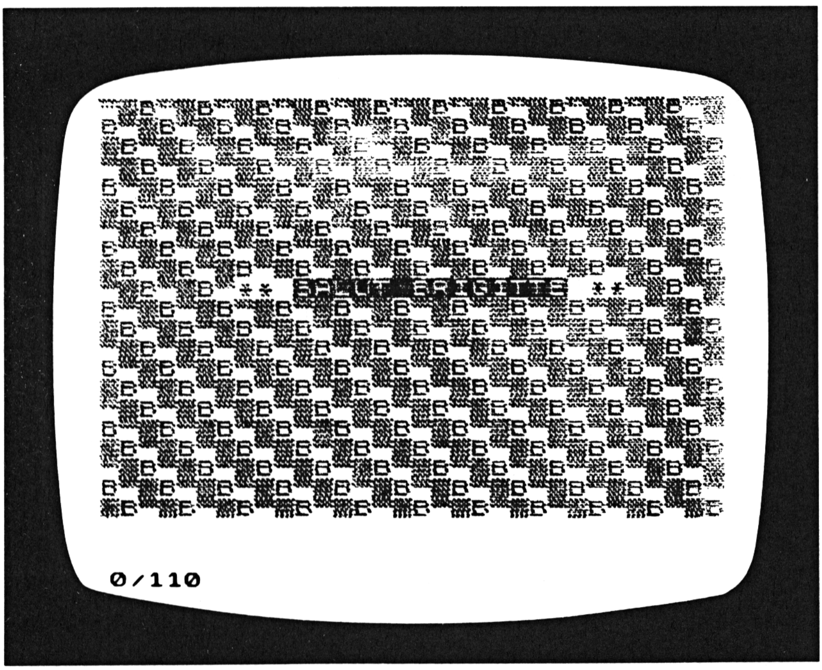


Figure 2.12 : Le Salut de l'ordinateur

Dans les chapitres à venir nous examinerons en détail le genre d'information que ce compte rendu d'exécution peut vous fournir. Dans notre cas, le zéro indique que le programme entier s'est terminé, et qu'aucun problème particulier n'a été rencontré pendant son déroulement. Le nombre 110 vous indique le numéro de la dernière ligne à avoir été exécutée.

ÉDITION D'UNE LIGNE QUELCONQUE DU PROGRAMME

Si vous avez fini d'admirer le salut amical de l'ordinateur, appuyez alors sur la touche NEW LINE (ou ENTER). Le message disparaît, et la liste du programme revient à l'écran. Vous pouvez exécuter le programme autant de fois que vous le voulez, toujours en introduisant le mot clé RUN; cela ne fatiguera jamais l'ordinateur.

Vous pouvez également modifier le programme si vous le souhaitez. Vous pouvez le faire de plusieurs façons, suivant le type de modification que vous voulez apporter :

- Vous pouvez ajouter une nouvelle ligne, simplement en la tapant au clavier après vous être assuré que son numéro de ligne n'a pas déjà été utilisé dans le programme.
- Vous pouvez supprimer une ligne quelconque en tapant son numéro de ligne, puis en faisant NEW LINE (ou ENTER). (Sachant cela, vous prendrez garde de ne pas supprimer accidentellement de ligne en introduisant un numéro déjà utilisé dans le programme.)
- Vous pouvez complètement réécrire une ligne entière en introduisant une nouvelle instruction avec le même numéro de ligne. L'ancienne ligne disparaîtra, et la nouvelle la remplacera.
- Enfin, si vous n'avez que peu de chose à modifier dans une ligne, vous pouvez l'*éditer.* Nous allons maintenant essayer de le faire.

Vous avez peut-être remarqué un léger détail que nous n'avons jusqu'à présent pas mentionné, dans la liste du programme : la

petite flèche en vidéo inversée qui pointe sur la dernière ligne introduite dans le programme. Si vous avez tapé les lignes du programme dans l'ordre, cette flèche pointe actuellement sur la ligne 110. (Vous pouvez voir la flèche à la Figure 2.1.) La flèche indique quelle est la *ligne courante* c'est-à-dire quelle ligne sera éditée si vous appuyez sur la touche EDIT. Vous pouvez déplacer la flèche vers le haut ou le bas de la liste du programme en appuyant sur les flèches Vers le haut et Vers le bas de votre clavier (SHIFT-6 et SHIFT-7).

Supposons, par exemple, que vous vouliez modifier quelque peu le message affiché à l'écran par l'ordinateur. Vous voulez remplacer le mot BONJOUR par BONSOIR. Cette partie du message se trouvant à la ligne 70, c'est celle-ci que vous souhaitez éditer.

Si le pointeur de ligne courante est sur la ligne 110, vous avez besoin de le déplacer de 4 lignes vers le haut pour éditer la ligne 70. Voici la marche à suivre :

1. Appuyez d'un doigt sur la touche SHIFT; puis appuyez 4 fois sur la touche 7. A chaque fois que vous appuyez, la liste est réécrite sur l'écran, et la flèche remonte d'une ligne.
2. Puis, la flèche pointant sur la ligne 70, appuyez sur la touche EDIT (SHIFT-1) et une copie de la ligne est reproduite au bas de l'écran, prête à être éditée.
3. En utilisant la touche flèche droite (SHIFT-8), déplacez le caractère indicateur L à droite de la lettre U de BONJOUR. Appuyez sur la touche RUBOUT (ou DELETE) trois fois pour supprimer trois lettres.
4. Passez en mode graphique (SHIFT-9) et introduisez les nouveaux caractères : S O I.
5. Enfin, quittez le mode graphique (à nouveau par SHIFT-9) et appuyez sur la touche NEW LINE (ou ENTER). La version modifiée de la ligne 70 apparaîtra dans la liste du programme.

Exécutez le programme maintenant pour être certain qu'il fonctionne toujours. Tout doit être identique, à l'exception de BONSOIR qui remplace BONJOUR dans le message.

Une remarque supplémentaire au sujet du déplacement de la

flèche de ligne courante dans un programme : quand vous travaillez sur un très long programme, vous pouvez désirer un moyen plus rapide pour déplacer la flèche d'une ligne à une autre, particulièrement si les lignes sont éloignées l'une de l'autre. Dans ce cas vous pouvez utiliser le mot clé LIST. Disons que vous voulez déplacer la flèche jusqu'à la ligne 40. Introduisez la commande :

```
LIST 40
```

Votre programme sera listé à l'écran à partir de la ligne 40. Puis appuyez sur la touche EDIT, et une copie de la ligne 40 est écrite au bas de l'écran pour être éditée.

LA DERNIÈRE CHOSE À FAIRE...

Au Chapitre 1, nous avons présenté la nécessité d'un magnétophone à cassettes comme moyen de stockage permanent des programmes. Vous vous souvenez que les programmes sont rangés temporairement dans la mémoire de l'ordinateur, et qu'ils sont perdus dès que l'on met l'ordinateur hors tension.

Dans ce paragraphe, vous apprendrez à utiliser un magnétophone à cassettes avec votre ordinateur. Si vous avez un magnétophone, préparez-le. Nous allons procéder à une séance pratique d'enregistrement, avec le programme de SALUT.

Sauvegarde de votre programme sur cassette

Prenez le petit cordon ayant une paire de fiches à chaque bout. S'il possède deux fiches, c'est pour vous permettre de relier à la fois les prises haut-parleur et micro de votre magnétophone à cassettes à votre ordinateur. En fait, c'est cependant plus sûr de ne brancher qu'un cordon à la fois. Les brancher simultanément peut parfois interférer avec le processus d'enregistrement. Les fiches sont de couleurs différentes afin de repérer plus facilement

quelle fiche à une extrémité correspond avec celle de l'autre extrémité.

Pour enregistrer un programme sur cassette à partir de l'ordinateur, relier d'abord la prise jack marquée "MIC" (sur le côté gauche de l'ordinateur) à la prise micro de votre magnétophone. Assurez-vous que les deux extrémités du cordon sont fermement enfichées dans les bonnes prises jack. (Si vous trouvez qu'il y a du jeu au niveau de la fiche de l'ordinateur, vous devrez peut-être maintenir la fiche pendant l'enregistrement et la relecture.)

Le réglage du volume et de la tonalité du magnétophone à cassettes est important. Commencez avec un volume réglé à peu près à la moitié du volume maximal. Réglez la tonalité au plus aigu et/ou au moins grave. Si votre premier essai d'enregistrement d'un programme ne réussit pas, essayez en augmentant le volume.

Installez une cassette vierge dans le magnétophone, et rembobinez-la si nécessaire. Si votre magnétophone est muni d'un compteur, remettez-le à zéro de façon à garder trace des points de début et de fin d'enregistrement.

Le mot clé à utiliser pour sauvegarder un programme sur cassette est SAVE, situé au-dessus de la touche S. La commande SAVE doit être suivie d'un nom de programme, tapé entre guillemets. Choisissez un nom de programme qui vous rappelle exactement ce que fait le programme. Nous appellerons ce programme SALUT. Entrez la commande, mais n'appuyez pas encore sur NEW LINE (ou ENTER) :

```
SAVE "SALUT"
```

Appuyez maintenant sur la touche d'enregistrement de votre magnétophone, laissez-le tourner quelques secondes, puis appuyez sur la touche NEW LINE (ou ENTER) de votre ordinateur. Votre écran va devenir blanc pendant quelques instants. Lorsque vous verrez l'écran se remplir de lignes noires horizontales (d'environ 2 cm d'épaisseur) bougeant rapidement en hauteur, vous saurez que le programme est en cours de transmission vers le magnétophone à cassettes. Le temps de présence des bandes sur l'écran dépend de la longueur du programme. Il ne faudra que quelques secondes pour enregistrer le programme SALUT. Pour de très longs programmes, cela peut prendre plusieurs minutes.

Une fois l'enregistrement terminé, l'écran redevient blanc, si ce n'est le message :

0/0

affiché dans le coin en bas à gauche.

Vous pouvez maintenant utiliser votre magnétophone pour écouter l'enregistrement si vous voulez être sûr que tout s'est bien passé. L'enregistrement d'un programme produit un bruit aigu, désagréable et strident, rappelant celui d'une sirène d'alarme. Vous n'avez pas besoin de l'écouter longtemps. Si vous avez entendu le début, arrêtez votre magnétophone; l'enregistrement est probablement réussi. (Si vous n'entendez pas ce bruit, vous devez recommencer l'essai d'enregistrement.)

Vous devez toujours faire deux copies de tout programme important, sur deux cassettes différentes. Mettez l'une d'entre elles en lieu sûr; elle sera votre copie de *sécurité* au cas où quelque chose arriverait à la copie que vous utilisez régulièrement. Sur chaque cassette, écrivez le nom du programme, et si possible, le numéro de sa position sur la cassette.

Pour rappeler un programme dans l'ordinateur depuis une cassette, vous devez relier la prise jack marquée "EAR" de votre ordinateur, à la prise de sortie de votre magnétophone. (A la prise de votre magnétophone, il peut être marqué "moniteur", "haut-parleur supplémentaire" ou "écouteur".) Une fois de plus il est très important de bien enfoncer les fiches.

La commande LOAD, située au-dessus de la touche J, est utilisée pour charger un programme dans l'ordinateur à partir d'une cassette. En ce qui concerne le programme SALUT il faut introduire la commande :

```
LOAD "SALUT"
```

Si vous ne connaissez pas le nom du programme, vous pouvez également entrer la commande sans nom, mais il faut taper le double guillemet :

```
LOAD ""
```

Assurez-vous que la cassette a été rembobinée au bon endroit, et appuyez sur la touche de lecture de votre magnétophone à

cassette. Puis appuyez sur la touche NEW LINE (ou ENTER) de votre ordinateur. Au début vous verrez en diagonale un motif de lignes noires, fines (de 3 à 7 mm environ) et assez stables sur l'écran. Puis au moment où le programme commencera, les mêmes bandes noires qu'à l'enregistrement apparaîtront. Quand sur la cassette on aura atteint la fin du programme, votre écran TV redeviendra blanc. A nouveau vous verrez le message :

```
0/0
```

dans le coin gauche de l'écran. Ceci indique le succès de la relecture du programme. Appuyez sur la touche NEW LINE (ou ENTER) et le programme sera listé à l'écran.

Si le motif des bandes ne s'est pas arrêté comme il aurait dû, c'est que quelque chose n'a pas fonctionné. Arrêtez le magnétophone, et appuyez sur la touche SPACE de votre ordinateur. (Remarquez que le mot BREAK est écrit au-dessus de la touche. Vous pouvez toujours appuyer sur BREAK pour interrompre l'exécution, le chargement ou la sauvegarde d'un programme.)

Les raisons les plus fréquentes d'échec en essayant de sauvegarder ou de charger un programme sont :

1. Des branchements incorrects. (Vérifiez que les fiches sont bien enfoncées dans les bonnes prises.)
2. De mauvais réglages de volume ou de tonalité. (Vous devez essayer de trouver le meilleur réglage du volume de votre magnétophone à cassettes.)

LES LOGICIELS DU COMMERCE

L'acquisition d'une parfaite maîtrise du système de lecture et d'écriture sur cassettes, vous permettant de sauvegarder et de charger des programmes facilement et avec fiabilité, peut exiger de vous une bonne dose de patience et de persévérance. Mais la récompense de vos efforts sera, en plus d'une sauvegarde permanente de vos programmes, la possibilité d'utiliser des programmes écrits et sauvegardés par d'autres que vous. En particulier, vous pourrez acheter des programmes – certains

d'entre eux étant assez sophistiqués – qui vous ouvriront de nouvelles perspectives d'utilisation de votre ordinateur.

La quantité de logiciels disponibles sur cassettes pour le ZX81 progresse rapidement. (En fait, on utilise le mot *logiciel (Software)* pour distinguer les programmes des composants physiques appelés le *matériel (Hardware)* qui constituent l'ordinateur.) Dans des domaines d'intérêt général tels que la gestion, les finances personnelles, l'éducation et les jeux, vous pourrez trouver un grand choix de programmes. Une grande partie de ces programmes nécessitent le module de mémoire supplémentaire de 16 K, mais quelques-uns peuvent être utilisés sans ce module.

Pour vous donner une idée du genre de programmes que vous pouvez acheter, nous verrons un exemple dans le prochain paragraphe de ce chapitre. Nous passerons en revue le programme VU-CALC, l'un des programmes les plus célèbres disponibles pour le ZX81.

VU-CALC, un programme " feuille de calculs "

Vous avez peut-être lu ou entendu parler de cette famille de logiciels appelée *programmes feuilles de calculs.* Ces programmes vous permettent de disposer de grandes quantités d'informations numériques, et de faire de nombreux calculs, avec rapidité et efficacité. De tels programmes sont maintenant disponibles sur la plupart des gros ordinateurs personnels. Deux exemples de tels programmes sont VisiCalc et SuperCalc, ces deux programmes ayant reçu un accueil phénoménal de la part des utilisateurs d'ordinateurs personnels. Les programmes *feuilles de calculs* simplifieront et rendront plus rapide tout travail de calculs nécessitant de garder trace des opérations, et de manipuler de grandes quantités de données dans des opérations arithmétiques.

VU-CALC, conçu pour le ZX81 muni de son module d'extension mémoire, est le "petit frère" de ces programmes. Bien qu'il ne soit pas aussi puissant ni universel que les programmes *feuilles de calculs* fonctionnant sur des ordinateurs plus gros, ce programme est un outil convenable pour tous ceux qui se trouvent confrontés à de lourdes et fastidieuses tâches de calcul arithmétique.

La Figure 2.13 présente un exemple simple d'une feuille de calculs VU-CALC. VU-CALC vous offre une grande feuille de travail de 26 lignes sur 36 colonnes; vous pouvez y inscrire des mots, des nombres ou des formules. Chaque position de cette feuille de travail est référencée par ses coordonnées (son *adresse*) : une lettre (indiquant la ligne) et un nombre (indiquant la colonne). Ainsi, par exemple, la position F01 de la feuille de calculs de la Figure 2.13 contient le mot NET. Les positions F02 et F03 comportent respectivement les recettes nettes des années 1981 et 1982.

Après avoir introduit des nombres sur votre feuille de calculs, vous pourrez créer de nouvelles données en écrivant sur la feuille des formules faisant référence aux nombres. Par exemple, les positions G02 et G03 montrent comment on a calculé le montant d'une taxe à partir des nombres contenus respectivement dans les cases F02 et F03. Dans le coin en bas à gauche de l'écran, vous

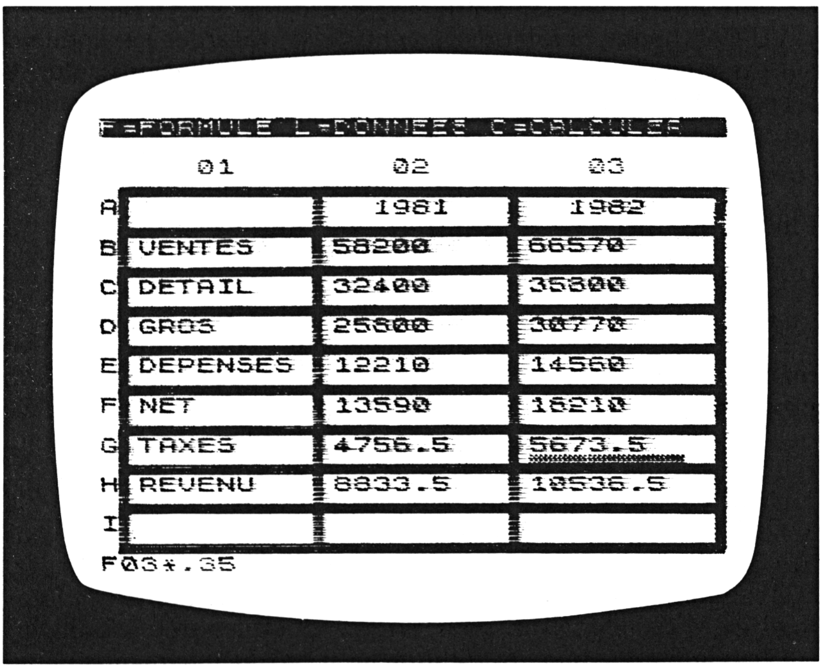


Figure 2.13 : Exemple du VU-CALC

pouvez voir la formule utilisée pour calculer la valeur se trouvant à la position G03 :

F03 * .35

(Dans cette formule, l'astérisque représente la multiplication. La formule apparaît à l'écran, car le *curseur* de la feuille de travail se trouve actuellement en G03.) Vous n'avez qu'à écrire de telles formules, et les appliquer, si vous le souhaitez, à de nombreuses lignes et colonnes de données. Le point le plus important est que de telles formules ont un caractère dynamique; elles refont les calculs dès qu'une donnée de la feuille de travail a été modifiée.

Comme vous ne pouvez voir que trois colonnes et neuf lignes de la feuille de travail à la fois, l'écran TV est utilisé comme une *fenêtre* que l'on pose au-dessus de la feuille tout entière. Pour visualiser les autres zones de la feuille de travail, vous pouvez *déplacer cette fenêtre* en utilisant les quatre touches de direction se trouvant en haut de votre clavier.

VU-CALC vous permet également de sauvegarder les données sur cassettes, à côté du programme. Cette méthode vous autorise à conserver de manière permanente les données de vos feuilles de travail importantes.

Un guide du consommateur de logiciels pour le ZX81

Le logiciel sur cassettes disponible pour votre ordinateur est malheureusement cher. Si vous achetez une demi-douzaine de programmes, vous aurez dépensé à peu près autant en logiciels que ce que vous aviez dépensé pour l'ordinateur. Pour cette raison, vous devez apprendre à devenir un consommateur avisé, sachant choisir et bien informé, lorsque vous vous déciderez à acheter du logiciel. Vous devez trouver un revendeur qui vous laisse expérimenter le programme avant que vous ne l'achetiez. (De nombreux revendeurs ont des *versions de démonstration* des logiciels qu'ils diffusent, vous pouvez donc les examiner.)

Chaque programme est fourni avec sa *documentation* – un mode d'emploi décrivant le fonctionnement du programme.

Prenez le temps de lire la documentation du programme que vous avez l'intention d'acheter. La documentation doit décrire clairement et simplement tout ce que vous avez besoin de savoir pour faire fonctionner et utiliser avec succès le programme. Si la documentation est mauvaise, vous pouvez perdre tellement de temps à essayer de comprendre le programme que votre achat ne vous profitera en rien.

Voici quelques-unes des questions que vous devez vous poser avant d'acheter un nouveau logiciel :

- Si le programme est orienté gestion ou finances personnelles, pourra-t-il être utilisé pour différentes applications ? Fait-il exactement ce dont vous avez besoin ? (En progressant dans la connaissance du BASIC, vous serez capable d'écrire vous-même de nombreux programmes, conçus exactement pour répondre à vos besoins.)
- Si le programme est un jeu, est-il de ceux avec lesquels vous prendrez du plaisir à jouer de nombreuses fois ?
- Si vous vous posez des questions sur l'utilisation du programme, serez-vous capable de trouver les réponses – soit dans la documentation, soit auprès de votre revendeur ? Quand vous essayez le programme dans la boutique, est-ce qu'il fait exactement ce qui est dit dans la documentation ?
- Comment réagit le programme quand vous faites une faute ? Vous fournit-il un moyen facile de correction des erreurs ? Vous devez faire volontairement des erreurs quand vous testez le programme, et noter ce qui se passe.

Vous poser de telles questions vous aidera à décider si tel ou tel programme mérite votre temps et votre argent.

RÉSUMÉ

L'apprentissage de l'utilisation du clavier de votre ordinateur est facile une fois que vous avez maîtrisé les différents modes du clavier. Puisque chaque ligne d'un programme BASIC doit commencer par un mot clé, l'ordinateur commence toujours par vous afficher le caractère indicateur K, ceci indiquant le mode K.

Après le mot clé, une instruction peut comporter des lettres, des nombres, ou des caractères majuscules (le mode L) ainsi que des graphismes ou des caractères inverses (le mode G), des fonctions (le mode F).

Votre ordinateur dispose de plusieurs moyens pour vous aider à détecter et corriger toutes les erreurs que vous pourriez faire en introduisant un programme. Si vous essayez d'introduire une ligne comportant une erreur de syntaxe, l'ordinateur rejettera cette ligne, et l'affichera en bas de l'écran, avec le caractère indicateur S pour indiquer la présence d'une erreur. Pour corriger cette ligne, vous pouvez utiliser les flèches gauche et droite pour déplacer le caractère indicateur à la bonne position de la ligne; puis, vous pouvez insérer ou supprimer des caractères. D'autre part, si vous introduisez une ligne correcte syntaxiquement, et que vous vouliez ensuite la modifier, vous pouvez utiliser la commande EDIT pour le faire.

La maîtrise de l'utilisation d'un lecteur-enregistreur à cassettes avec votre ordinateur peut vous demander un certain nombre d'essais jusqu'à ce que cela se fasse sans erreurs. Cependant, une fois que vous saurez sauvegarder et charger des programmes, votre ordinateur deviendra un outil bien plus précieux. Vous serez capable de commencer à apprécier les avantages respectifs des logiciels du commerce, et vous pourrez commencer à vous constituer une bibliothèque de programmes que vous aurez écrits.

Chapitre 3

L'AFFAIRE SE CORSE : UN RAPIDE COURS SUR LES GRAPHISMES EN BASIC

INTRODUCTION

Dans ce chapitre, vous étudierez les principes du BASIC, tout en vous amusant avec les possibilités graphiques de votre ordinateur. Naturellement, les techniques de programmation que vous acquerrez à la lecture de ce chapitre ne seront pas restreintes à la seule écriture de programmes graphiques. Vous utiliserez les mêmes instructions et les mêmes méthodes ensuite quand vous commencerez à écrire d'autres sortes de programmes. Mais les exercices graphiques de ce chapitre constituent un moyen attrayant pour commencer l'étude du BASIC. A chaque étape vous pourrez *visualiser* exactement ce que fait l'ordinateur, en réponse à vos ordres.

Le ZX81 est muni de deux systèmes distincts de création graphique à l'écran. Vous avez vu un exemple de l'un d'entre eux au Chapitre 2, avec l'utilisation de caractères graphiques spéciaux et l'affichage de caractères en vidéo inversée. Dans ce chapitre,

nous commencerons par examiner en détail les deux systèmes. Puis nous suivrons rapidement un petit cours de BASIC exposant les instructions que vous utiliserez le plus fréquemment dans vos programmes. Vous verrez des utilisations de chacune des instructions, dans plusieurs petits programmes, puis, pour résumer ce que vous aurez appris, vous travaillerez sur un bref programme graphique, traçant des rectangles à l'écran.

Le dernier programme de ce chapitre est assez long, puisqu'il occupe 2 K de la mémoire de votre ordinateur. (Si vous utilisez le ZX81, vous devrez donc posséder l'extension mémoire de 16 K.) Vous pourrez utiliser ce programme pour dessiner toutes sortes de figures et de dessins à l'écran. Ceux-ci deviendront meilleurs au fur et à mesure que votre expérience de ce programme progressera. Le programme lui-même mérite d'être soigneusement étudié, car il met en évidence plusieurs techniques importantes qui vous aideront à écrire des programmes plus clairs et plus performants.

LES COMMANDES EN MODE IMMÉDIAT

Au Chapitre 2 nous avons décrit trois des caractéristiques d'un programme BASIC :

- les lignes sont numérotées;
- chaque ligne commence par un mot clé;
- l'ordinateur se contente simplement de stocker les lignes, sans les exécuter, jusqu'au moment où vous taperez RUN.

Vous pouvez utiliser un certain nombre de mots clés (mais pas tous), en tant que commandes en *mode immédiat.* C'est-à-dire que vous pouvez taper une instruction, sans qu'elle soit précédée d'un numéro de ligne; l'ordinateur exécutera cette instruction immédiatement, sans attendre que vous ayez fait RUN. Parfois vous voudrez utiliser des commandes en mode immédiat pour explorer les réactions de l'ordinateur à une instruction ou à une autre.

Par exemple, introduisez la commande suivante au clavier :

```
PRINT "ZX81"
```

Vous verrez alors l'ordinateur écrire immédiatement ZX81 en haut de l'écran. Puis, si vous appuyez à nouveau sur NEW LINE (ou ENTER) vous verrez que l'ordinateur n'a pas sauvegardé cette instruction PRINT. Les commandes en mode immédiat sont exécutées une seule fois, puis elles sont perdues.

Vous avez déjà vu plusieurs autres exemples de commandes en mode immédiat. Vous savez que SAVE et LOAD sont utilisées pour ranger et changer des programmes sur cassette. LIST demande à l'ordinateur d'afficher les lignes du programme à l'écran. RUN fait exécuter à l'ordinateur les lignes du programme. Une autre commande que vous avez déjà pu utiliser est la commande NEW qui efface le programme actuellement en mémoire dans l'ordinateur afin de pouvoir écrire un nouveau programme. Toutes ces instructions sont habituellement utilisées en tant que commandes en mode immédiat, bien qu'elles puissent être utilisées en tant que lignes de programme.

D'autres commandes, telles que PRINT, sont habituellement utilisées comme instructions de programme, mais peuvent être exécutées en tant que commandes en mode immédiat. Nous utiliserons PRINT comme commande en mode immédiat dans le prochain paragraphe de ce chapitre pour mettre en pratique plusieurs expérimentations rapides.

LES DEUX SYSTÈMES GRAPHIQUES

L'instruction PRINT AT

Quand vous utilisez l'instruction PRINT pour afficher une information à l'écran, chaque caractère occupe la même place. Imaginez l'écran comme une grille, divisée en lignes et colonnes; vous pouvez placer un caractère dans chaque case de la grille. La zone d'écran possède 22 lignes en hauteur et 32 colonnes en largeur. (La zone du bas de l'écran, en dessous de la 22ème ligne, est une zone de travail réservée à l'ordinateur. Comme vous commencez à le savoir, c'est dans cette zone qu'apparaissent les nouvelles lignes que vous introduisez. Vous ne pouvez en aucun cas accéder à cette zone par l'instruction PRINT.)

L'utilisation de l'instruction PRINT AT permet de préciser exactement l'endroit où vous souhaitez que l'information apparaisse à l'écran. Comme vous l'avez vu au Chapitre 2, PRINT AT est toujours suivie de deux nombres. Ces nombres représentent une *adresse* sur l'écran. La Figure 3.1 vous aidera à comprendre comment on détermine ces adresses. Le premier nombre qui suit PRINT AT est le numéro de *ligne,* le second le numéro de colonne. Ainsi le format valide de l'instruction PRINT AT est :

PRINT AT ligne, colonne; "une information quelconque"

Remarquez une fois encore que les nombres précisant la ligne et la colonne sont séparés par une virgule, et que l'adresse sur l'écran est séparée de l'information par un point-virgule.

Regardez à nouveau la Figure 3.1. Les adresses partent du coin en haut à gauche de l'écran. Vous pouvez considérer ce point



Figure 3.1 : Utilisation de PRINT AT

comme l'*origine,* si vous voulez. Son adresse est 0,0. Remarquez les adresses des trois autres coins de l'écran. Le numéro de ligne augmente au fur et à mesure que vous descendez dans l'écran, il en est de même pour le numéro de colonne quand vous progressez vers la droite.

Pour être certain d'avoir compris ce système d'adresses, examinez les instructions suivantes, en essayant de trouver approximativement à quels endroits sur l'écran elles afficheront les messages :

```
PRINT AT 19,5; "X"
PRINT AT 7,7; "BONJOUR"
PRINT AT 15,25; "AU REVOIR"
```

Entrez maintenant ces commandes, une à une, et voyez si ce que vous aviez prévu se passe bien. Remarquez que lorsque le message comporte plus d'un caractère, c'est le *premier* caractère qui est placé à l'endroit spécifié dans l'adresse. Les autres caractères viennent à la suite, sur la même ligne.

Vous pouvez utiliser une seule instruction PRINT pour afficher plusieurs messages à des endroits différents. Essayez la commande suivante :

```
PRINT AT 3,3; "ICI"; AT 20,25; "ET LA"
```

En bref, on peut utiliser PRINT AT pour afficher n'importe quel caractère, mot, ou caractère graphique spécial, en n'importe quelle position de l'écran 22 sur 32.

PLOT et UNPLOT

Votre ordinateur vous offre aussi une deuxième méthode pour tracer des graphismes sur l'écran TV, ce sont les mots clés PLOT et UNPLOT. Ces instructions contrôlent l'écran quelque peu différemment et utilisent un nouveau système d'adresses.

La Figure 3.2 résume la façon d'utiliser PLOT et UNPLOT. L'instruction PLOT inscrit un petit carré blanc, que l'on appelle un *pixel* (c'est-à-dire un point graphique) à l'adresse spécifiée, sur l'écran. Chaque exécution de PLOT place un seul pixel à l'écran.

Les quatre coins de l'écran à la Figure 3.2 comportent un pixel. On peut afficher sur l'écran 44 lignes de 64 colonnes de pixels. Souvenez-vous du système d'adresses de l'instruction PRINT AT. Si vous divisez chaque case de la grille de PRINT AT en quatre petits carrés, vous retrouvez la dimension d'un pixel.

L'origine des adresses en pixels n'est pas le coin en haut à gauche de l'écran, mais le coin en bas à gauche. Cette position possède l'adresse 0,0. Les coordonnées des lignes progressent jusqu'à 43 en montant vers le haut de l'écran, et celles des colonnes jusqu'à 63 en se déplaçant vers la droite. La forme générale de l'instruction PLOT est :

```
PLOT colonne, ligne
```

Pour vous familiariser avec les pixels et le système d'adresses qu'utilise l'instruction PLOT, introduisez les lignes suivantes, une



Figure 3.2 : Utilisation de PLOT et UNPLOT

à une, en tant que commandes en mode immédiat, et faites attention à l'endroit où chaque pixel se place :

```
PLOT 5,5
PLOT 55,3
PLOT 40,40
```

L'instruction UNPLOT *efface* un pixel de l'écran. Elle a la même forme que l'instruction PLOT :

```
UNPLOT colonne, ligne
```

Nous verrons de nombreuses utilisations des instructions PRINT AT, PLOT et UNPLOT, dans les programmes de ce chapitre, aussi soyez sûr d'avoir bien compris leur fonctionnement avant d'aborder ces programmes. Si vous hésitez encore un peu au sujet de ces deux systèmes d'organisation de l'écran, revoyez les Figures 3.1 et 3.2, et essayez des commandes en mode immédiat.

UN RAPIDE COURS DE BASIC

Ce paragraphe vous présente l'essentiel de la programmation en BASIC du ZX81 : définir des variables, lire une information en entrée, répéter des instructions, contrôler la séquence d'exécution et prendre des décisions. Vous trouverez de nombreux et courts exemples de programmes, illustrant chaque point. Il est important que vous entriez tous ces programmes dans votre ordinateur, et que vous les testiez. Si vous ne comprenez pas pourquoi un résultat se produit, revenez en arrière et revoyez les explications données.

Retour sur les variables

Nous avons vu qu'une variable est tout simplement une position dans la mémoire de l'ordinateur, réservée à un certain type d'information. Il vous est possible de définir de nombreuses variables différentes, pour les utiliser dans le programme. Afin de faciliter l'accès à chaque information, toutes les variables ont un

nom. L'information contenue dans une variable peut changer, mais son nom reste le même. Nous avons déjà vu qu'il y a deux *types* de variables – les variables numériques et les variables de texte appelées *chaînes de caractères.*

Vous pouvez définir des variables, ou modifier leur valeur, grâce à l'instruction LET. La forme générale de l'instruction LET est :

```
LET NOM = VALEUR
```

Cet ordre signifie "ranger VALEUR dans la variable dont le nom est NOM". Quand l'ordinateur exécute une instruction LET, il évalue d'abord l'information placée à droite du signe égal, puis range cette information dans la variable dont le nom est à gauche du signe égal.

Voyons quelques exemples. L'instruction LET peut être utilisée en tant que commande en mode immédiat, vous pouvez donc introduire les lignes suivantes directement, sans numéros de lignes. Supposons que vous vouliez définir une variable nommée REVENU pour stocker le montant d'un revenu que vous avez touché cette année. Votre commande peut être quelque chose du genre :

```
LET REVENU = 15000
```

Introduisez cette commande et voyez le résultat. Vous ne voyez que le message 0/0 dans le coin en bas à gauche de l'écran – signifiant que l'ordinateur a exécuté la commande.

Maintenant, pour vous assurer que REVENU a bien été défini, faites :

```
PRINT REVENU
```

Vous verrez que la *valeur* de la variable REVENU – 15000 – est affichée en haut de l'écran. Ainsi, quand vous ajoutez le nom d'une variable à l'instruction PRINT, l'ordinateur affiche à l'écran la *valeur* contenue dans cette variable.

Vous êtes libre de choisir n'importe quel nom pour une variable. Chaque fois que c'est possible, il est préférable de choisir un nom décrivant ce que la variable représente. Dans le cas de variables *numériques,* c'est aussi à vous de décider la

longueur du nom de la variable. Certains préfèrent utiliser des noms de variables longs, tels que REVENU, car cela facilite la lecture du programme et ils voient ainsi directement à quoi sert la variable. Les inconvénients de noms de variables longs sont, d'une part, qu'ils demandent davantage de temps à être introduits au clavier, et d'autre part qu'il prennent davantage de place dans la mémoire de l'ordinateur. Pour ces raisons, vous pouvez trouver vous-même des noms de variables plus courts, des abréviations, telles que REV ou simplement R.

Un fois définie une variable par un certain nom, l'ordinateur ne reconnaîtra que le nom spécifié. Si vous vous trompez accidentellement de nom de variable, ou si vous essayez une abréviation alors que le nom a déjà été défini, l'ordinateur ne comprendra pas ce que cela signifie. Par exemple, maintenant que vous avez défini la variable REVENU, essayez d'entrer chacune des lignes suivantes :

```
PRINT RIVENU
PRINT REV
PRINT R
```

Aucun de ces noms ne peut être utilisé pour accéder à l'information contenue dans la variable REVENU. Après chacune de ces commandes, vous verrez le message :

```
2/0
```

dans le coin en bas à gauche de l'écran. Le 2 est un code qui représente l'un des messages d'erreur de l'ordinateur; il signifie simplement que vous avez utilisé un nom de variable non défini. (Vous avez les codes de tous les messages d'erreur à l'Appendice B.)

Dans un programme les variables représentent le moyen de stocker les éléments d'information dont vous aurez besoin à nouveau. En plus des informations numériques, vous pouvez également vouloir stocker des caractères, des mots ou du texte dans la mémoire de l'ordinateur. Comme nous l'avons vu, des éléments d'information *non numérique* sont appelés des chaînes de caractères, et les variables les contenant sont appelées *variables chaînes de caractères.* Le nom d'une variable chaîne de

caractères, vous vous en souvenez, doit se terminer par le caractère "$". Contrairement aux noms de variables numériques, les noms de variables chaînes de caractères doivent être exactement de deux caractères : une lettre, puis le caractère $.

Voici quelques exemples d'instructions LET, définissant des variables chaînes de caractères :

```
LET A$ = "18 RUE LECOURBE "
LET C$ = "PARIS "
LET S$ = "15EME "
```

Introduisez ces trois ordres LET, puis tapez la commande :

```
PRINT A$; C$; S$
```

Le résultat sera :

```
18 RUE LECOURBE PARIS 15EME
```

affiché en haut de l'écran. Ceci vous montre quelque chose de nouveau par rapport à l'instruction PRINT; elle peut contenir autant de noms de variables que vous le souhaitez. L'ordinateur affichera simplement le contenu de chacune des variables à l'écran. Remarquez que toutes les variables sont séparées par un point-virgule dans l'instruction PRINT. Ceci indique à l'ordinateur d'afficher le contenu des variables côte à côte sur la même ligne.

Maintenant essayez ces deux ordres PRINT :

```
PRINT "JE VIS A "; C$; "."
PRINT "MON REVENU EST DE "; REVENU; " F."
```

Ces lignes montrent qu'une instruction PRINT peut comporter n'importe quelle combinaison d'éléments : messages littéraux, variables numériques, variables chaînes de caractères. La première de ces deux commandes provoquera le message :

```
JE VIS A PARIS
```

en haut de l'écran. Que fait la seconde ? Comme expérimentation supplémentaire, utilisez la commande LET pour changer la valeur de C$ par le nom de la ville où vous habitez, et la valeur de REVENU par le montant de votre revenu. Puis introduisez les deux commandes ci-dessus une seconde fois, et regardez les résultats.

Il y a plusieurs façons de supprimer de la mémoire de l'ordinateur des variables et les informations qu'elles contiennent. La plus simple, naturellement, est de couper le courant, mais si vous le faites vous perdrez tout, y compris le programme sur lequel vous avez pu travailler. Autrement vous pouvez utiliser le mot clé :

```
CLEAR
```

situé au-dessus de la touche X. Cette commande efface toutes les variables et les informations en mémoire, mais laisse intactes toutes les lignes du programme. Introduisez maintenant cette commande, puis tapez :

```
PRINT REVENU
```

Vous obtiendrez le message d'erreur 2/0, qui vous dit que la variable REVENU n'a pas été définie.

Pour terminer, chaque fois que vous exécuterez un programme, en utilisant le mot clé RUN, les variables et les informations placées en mémoire au cours des exécutions précédentes seront effacées de la mémoire.

Une seconde façon de définir des variables, en plus de l'instruction LET, est l'instruction INPUT. Etudions maintenant cette instruction.

LECTURE D'INFORMATIONS DEPUIS LE CLAVIER

Sur le ZX81 l'ordre INPUT n'a qu'une seule forme :

```
INPUT nom
```

où "nom" peut être un nom quelconque de variable.

Comme vous l'avez vu dans le programme SALUT du Chapitre 2, INPUT dit à l'ordinateur d'attendre qu'une information ait été frappée au clavier, puis de la ranger dans la variable dont le nom se trouve dans l'ordre INPUT. Cette variable peut aussi bien ne pas avoir encore été définie, que posséder déjà une valeur.

Dans ce dernier cas, l'ancienne valeur est perdue et la nouvelle valeur, lue au clavier, la remplace.

INPUT ne peut pas être utilisée en tant que commande en mode immédiat, aussi nous écrirons un petit programme pour expérimenter cette instruction. Entrez les six lignes suivantes :

```
10 PRINT AT 21,0; "NOM ? "
20 INPUT N$
30 PRINT AT 21,0; "AGE ? "
40 INPUT A
50 CLS
60 PRINT N$; ",AGE ";A
```

En exécutant ce programme, vous verrez que son déroulement s'interrompra deux fois pour que vous puissiez introduire les informations au clavier. D'abord tapez votre nom, puis votre âge. Une fois que vous aurez fini, un message du genre de celui-ci apparaîtra en haut de l'écran :

```
JEAN DUPONT, AGE 31
```

Ce programme et ce qui en résulte peut vous apprendre plusieurs choses importantes. Tout d'abord, ce que l'ordinateur affiche sur l'écran lorsqu'il attend l'introduction d'une information au clavier : si vous avez regardé attentivement l'écran pendant l'exécution du programme, vous aurez remarqué deux sortes différentes de messages. Quand l'ordinateur attend une chaîne de caractères, il affiche le caractère indicateur L (en vidéo inversée, naturellement) entre guillemets. Par contre, quand l'information attendue est numérique, il n'affiche que le seul caractère L. C'est de cette façon que l'ordinateur fait la distinction entre un ordre INPUT contenant un nom de variable chaîne de caractères ou un nom de variable numérique.

Les lignes 10 et 30 de ce petit programme présentent un point important dans la conception d'une programmation claire et efficace. Toutes les fois que vous écrirez un programme utilisant des ordres INPUT, vous devrez songer soigneusement à la personne qui utilisera ce programme. Cette personne aura besoin qu'on lui dise quelle information elle doit introduire dans l'ordinateur, au moment où l'exécution s'interrompra pour une

entrée. Pour cette raison, afficher sur l'écran un "message indicateur" chaque fois que l'ordinateur est dans l'attente d'une entrée est une bonne idée. Les lignes 10 et 30 donnent un exemple de ce genre de "messages indicateurs". Remarquez que ces lignes positionnent leur message tout près du bas de l'écran. C'est pour que ce message apparaisse autant que possible, comme un " écho " à l'information en train d'être introduite au clavier.

La ligne 50 de ce programme remet à blanc l'écran, et ainsi le message de la ligne 60 apparaîtra en haut de l'écran. Etudiez soigneusement la ligne 60; remarquez que le message entre guillemets (" ,AGE ") doit comporter une virgule et deux espaces, pour séparer les informations des deux variables :

```
60 PRINT N$; " ,AGE "; A
```

Résumons-nous : nous avons vu que l'on peut définir les variables de deux façons, avec l'instruction LET ou l'instruction INPUT. En général vous n'utiliserez que des variables définies de l'une de ces deux façons. (Il y a une exception à cette règle, que nous verrons dans le prochain paragraphe de ce chapitre.) Une fois qu'une variable a été définie, et qu'on lui a donné une valeur, vous pouvez utiliser l'ordre PRINT pour afficher cette valeur à l'écran. Vous pouvez également *modifier* la valeur contenue dans une variable par de nouveaux ordres LET et INPUT. Nous continuerons l'étude de ce sujet dans la suite du chapitre.

Avant de continuer, voyons un programme encore plus court illustrant l'usage des variables. Tapez le mot clé NEW pour effacer le dernier programme de la mémoire de l'ordinateur. Puis tapez les cinq lignes suivantes :

```
200 PRINT AT 21,0; "HORIZONTAL"
210 INPUT H
220 PRINT AT 21,0; "VERTICAL"
230 INPUT V
240 PLOT H,V
```

Ces lignes font réellement partie d'un programme plus long que nous construirons plus loin, mais vous pouvez exécuter ces cinq lignes comme si elles constituaient un programme complet.

Quand vous exécuterez ce programme, vous verrez deux messages indicateurs d'entrée à l'écran, d'abord :

```
HORIZONTAL
```

à cette étape vous devrez introduire un nombre compris entre 0 et 63; puis :

```
VERTICAL
```

et là vous devrez, en réponse, introduire un nombre compris entre 0 et 43. Une fois que vous aurez introduit les deux nombres, le programme affichera à l'écran un pixel à l'adresse que vous aurez spécifiée par les deux nombres.

Assurez-vous de bien comprendre le fonctionnement du programme. Les lignes 210 et 230 comportent les instructions INPUT, définissant respectivement les variables H et V. Puis la ligne 240 utilise les valeurs de H et V en tant qu'adresse dans l'instruction PLOT :

```
240 PLOT H,V
```

Les instructions PLOT et UNPLOT ne nécessitent pas de nombres littéraux pour l'adresse du pixel. Elles peuvent utiliser des variables comme adresse, du moment que ces variables ont été définies et comportent une adresse de pixel correcte.

Nous allons maintenant étudier les ordres FOR et NEXT; souvenez-vous que nous les avons déjà vus au Chapitre 2, et qu'on peut les utiliser pour faire répéter un certain nombre de fois des instructions par l'ordinateur.

LES INSTRUCTIONS DE RÉPÉTITION

Voici un exemple de la forme la plus simple de l'instruction FOR :

```
FOR I = 1 TO 20
```

I dans cette instruction peut être appelée *variable de contrôle* car elle contrôle le nombre de répétitions de l'ordre FOR et des

ordres suivants. Le nom de la variable de contrôle ne doit avoir qu'une lettre. Une variable de contrôle n'a pas besoin d'être définie à l'avance par une instruction LET ou INPUT. (C'est l'exception à la règle.)

Examinons le fonctionnement de l'instruction FOR. FOR et NEXT ne peuvent pas être utilisées en tant que commandes en mode immédiat, nous allons donc les mettre en œuvre dans un petit programme :

```
10 FOR I = 1 TO 20
20 PRINT "REPETITION"
30 NEXT I
```

Souvenez-vous que les lignes FOR et NEXT représentent le haut et le bas de la *boucle.* Toutes les lignes intermédiaires seront répétées le nombre de fois spécifié. Voici comment l'ordinateur

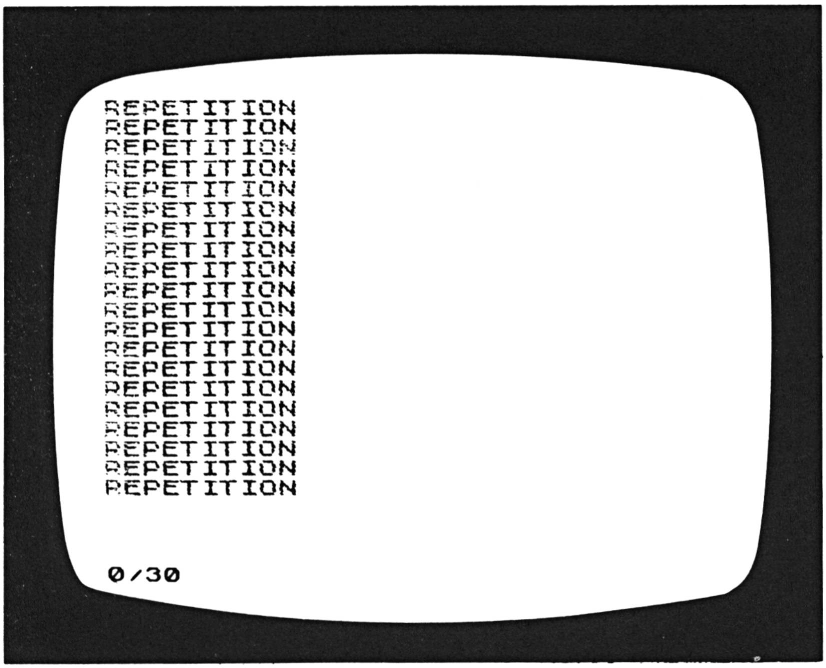


Figure 3.3 : Répétition

utilise la variable de contrôle, I dans notre cas, pour décider du nombre de répétitions :

1. Au début, on donne à I la valeur qui se trouve juste derrière le signe égal : 1 dans notre cas.
2. Toutes les instructions situées entre FOR et NEXT sont exécutées.
3. Quand l'ordinateur arrive à la ligne NEXT, il augmente (ou *incrémente*) de 1 la valeur de I, puis retourne à la ligne FOR.
4. Si I est encore *plus petit ou égal* au nombre spécifié après le mot TO (20 dans notre cas), l'ordinateur répète toutes les lignes jusqu'à NEXT une nouvelle fois. Si I est plus grand que le nombre apparaissant derrière TO, alors la boucle s'arrête.

Introduisez ces trois lignes dans votre ordinateur, et exécutez-les. La Figure 3.3 montre le résultat. Comme vous pouvez le voir, l'instruction PRINT de la ligne 20 a été exécutée 20 fois. Autre-



Figure 3.4 : Répétition encore

ment dit, la variable de contrôle I a été incrémentée de 1 à 20; après chaque incrémentation l'instruction PRINT a été exécutée une fois.

Pour explorer plus profondément le concept de répétition, nous verrons deux variantes de ce petit programme, chacune d'elles apportant une légère modification à la ligne 20. Pour effectuer ces modifications, utilisez la commande EDIT de votre ordinateur.

D'abord, modifiez la ligne 20 de la façon suivante :

```
20 PRINT "REPETITION ";
```

Deux modifications ont été faites : un espace a été ajouté entre le N et le deuxième guillemet, et un point-virgule a été ajouté en fin de ligne. Une fois que vous aurez fait ces modifications, exécutez à nouveau le programme; le résultat sera tel qu'à la Figure. 3.4. Ce petit exercice a simplement pour objet de vous

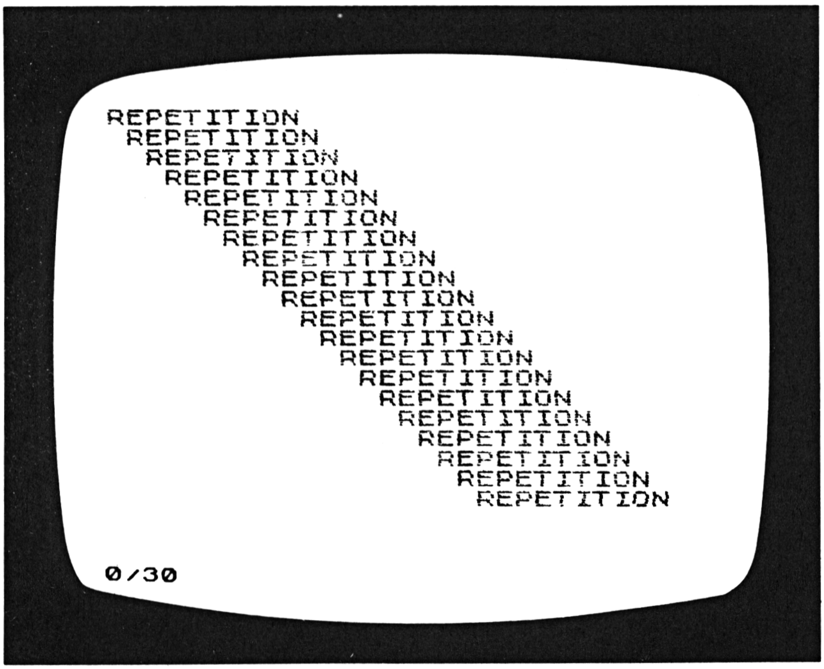


Figure 3.5 : Répétition toujours...

rappeler le rôle du point-virgule dans l'instruction PRINT. Le point-virgule indique à l'ordinateur de *ne pas passer* à la ligne suivante. (Nous pouvons dire que le point-virgule évite un *saut à la ligne.*) De cette manière, lorsqu'une instruction PRINT *se termine* par un point-virgule, l'instruction PRINT suivante est exécutée en repartant de la fin du message précédent.

Comme dernier essai, modifions ainsi la ligne 20 :

```
20 PRINT AT I,I; "REPETITION"
```

et exécutons le programme une troisième fois. On voit le résultat à la Figure 3.5.

Cette dernière version du programme met en évidence un point essentiel : *La variable de contrôle de l'instruction FOR peut être utilisée en tant que variable, dans les autres types d'instructions apparaissant à l'intérieur de la boucle.* Le fait que la variable de contrôle reçoive une valeur incrémentée à chaque fois que l'on répète la boucle, peut faire de cette variable un outil très précieux pour certains types de tâches répétitives. A la Figure 3.5, vous pouvez voir la variable de contrôle I, déterminer l'adresse de chaque occurrence de l'instruction PRINT AT. Comme la valeur de I progresse de 1 à 20, l'instruction PRINT AT est exécutée comme :

```
PRINT AT 1,1; "REPETITION"
PRINT AT 2,2; "REPETITION"
PRINT AT 3,3; "REPETITION"
PRINT AT 4,4; "REPETITION"
```

... et ainsi de suite.

Continuons l'examen de ce dernier point un peu plus à fond, avec un autre exemple. Modifiez ainsi votre programme de trois lignes :

```
10 FOR I = 20 TO 40
20 PLOT I,I
30 NEXT I
```

Dans cette version, nous utiliserons la valeur de la variable de contrôle pour déterminer les adresses des pixels que produira l'instruction PLOT. Exécutez le programme, et vous verrez se tracer une ligne diagonale de pixels, de l'adresse 20,20 à l'adresse

40,40. Remarquez que la variable de contrôle, I, ne commence pas à 1 dans ce programme : elle est incrémentée de 20 à 40, comme spécifié dans l'instruction FOR.

Si vous changez la ligne 20 en :

```
20 PLOT I,20
```

vous pouvez produire une ligne horizontale de pixels. De la même façon, la ligne :

```
20 PLOT 20,I
```

produira une ligne verticale de pixels, quand vous exécuterez le programme.

Dans les exemples vus jusqu'ici, la variable de contrôle a été incrémentée de 1 à chaque répétition de la boucle FOR. Dans un certain nombre de cas, vous voudrez mettre le *pas d'incrémentation* à une valeur différente de 1. Vous pouvez le faire avec le mot STEP; par exemple :

```
FOR I = 2 TO 40 STEP 2
```

(STEP est un caractère majuscule, situé sur la touche E.)

Pour voir le fonctionnement de STEP, étudions la version suivante de notre programme :

```
10 FOR I = 2 TO 40 STEP 2
20 PLOT I,20
25 PLOT 20,I
30 NEXT I
```

Dans ce programme, la variable de contrôle, I, prendra les valeurs 2, 4, 6, 8, ..., 40. Comme résultat, l'instruction PLOT produira des lignes en pointillés, car une adresse sur deux de la ligne aura été sautée. (Ce programme montre aussi la facilité avec laquelle on peut ajouter des lignes dans une boucle FOR. La ligne 25, seconde instruction PLOT, produit une deuxième ligne à l'écran.) Exécutez le programme. La Figure 3.6 montre ce qui apparaît à l'écran. Etudiez soigneusement ce programme et ses résultats, avant d'aller plus loin. Assurez-vous d'avoir compris tout ce qu'il fait.

Dans tous les programmes examinés jusqu'ici, l'ordre dans

lequel les lignes étaient traitées, n'était déterminé que par les numéros de lignes. L'ordinateur commençait par la ligne de numéro minimum, et progressait, ligne à ligne, jusqu'à ce que la dernière ligne du programme ait été exécutée. Parfois c'est une bonne méthode de conception d'un programme, mais on est souvent amené à exécuter certaines lignes dans un ordre différent. Nous examinerons pourquoi, et verrons dans le paragraphe suivant avec quels mots clés cela est possible.

L'organisation du fonctionnement

Le mot clé GOTO (situé au-dessus de la touche G) peut être utilisé pour dire à l'ordinateur d'exécuter une ligne hors-séquence. GOTO est simplement suivi du numéro de la ligne que vous voulez que l'ordinateur exécute ensuite. L'instruction GOTO

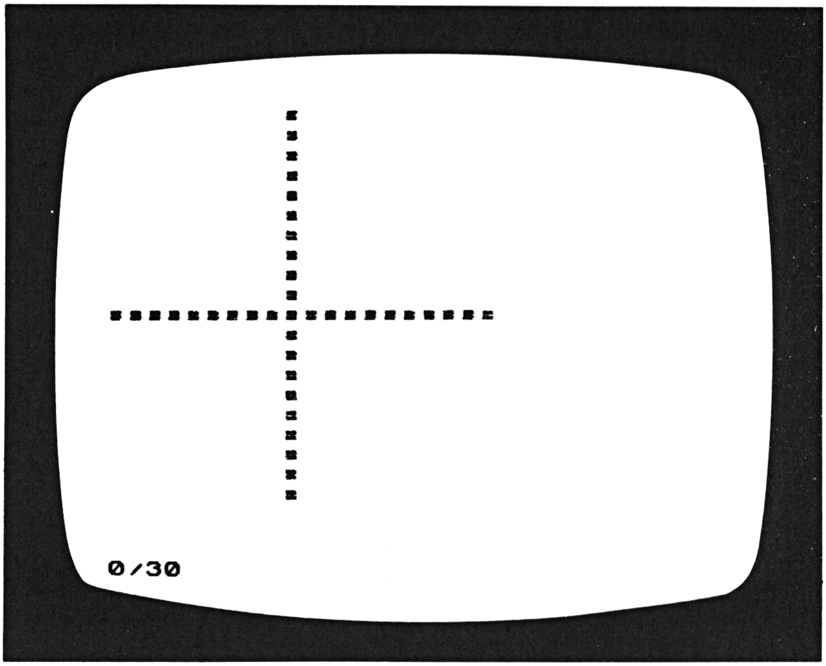


Figure 3.6 : Fonctionnement de STEP

peut passer le contrôle du programme à une ligne, plus avant dans le programme, dont le numéro est plus grand que celui de la ligne courante :

```
90 GOTO 300
```

ou passer le contrôle, plus en arrière dans le programme, à une ligne de numéro plus petit :

```
90 GOTO 10
```

L'instruction GOTO est une instruction simple, mais les raisons de l'utiliser sont souvent complexes. En général c'est une bonne idée de ne l'utiliser que modérément dans un programme. En effet, un abus d'instructions GOTO peut rendre les résultats de votre programme presque impossibles à prévoir avec certitude. Mais dans un certain nombre de situations l'emploi de GOTO ne peut être évité.

Dans les deux principaux programmes de ce chapitre, nous utilisons l'instruction GOTO pour créer une *boucle sans fin*. Le programme, ou une certaine partie du programme, sera répété sans cesse; le programme lui-même ne dispose d'aucune clause pour s'arrêter. Votre clavier vous fournit un moyen d'interrompre artificiellement un tel programme. Vous avez simplement à appuyer sur la touche BREAK, située dans le coin en bas à droite du clavier.

Les lignes suivantes sont un exemple de boucle sans fin :

```
10 PRINT AT 10,13; "ECLAIR"
20 PRINT AT 10,13; "          "
30 GOTO 10
```

Vous pouvez suivre l'ordre de déroulement du programme avant de l'exécuter. D'abord, les instructions PRINT des lignes 10 et 20 sont exécutées, puis la ligne 30 rend le contrôle au début du programme. Le processus continue sans interruption. Exécutez le programme pour voir le résultat. Ceci est bien sûr un exemple plutôt futile; des programmes ultérieurs appliqueront l'instruction GOTO à un meilleur usage.

Une autre instruction faisant passer l'ordinateur à un nouvel endroit du programme, en rompant avec la séquence des

instructions, est GOSUB. Cette instruction signifie "Aller à un sous-programme". Comme l'instruction GOTO, elle est suivie d'un numéro de ligne, par exemple :

```
GOSUB 200
```

Pour comprendre l'instruction GOSUB, vous devez savoir ce que sont des sous-programmes, et à quoi ils servent. Souvent, dans un programme qui accomplit nombre de tâches différentes, certaines de ces tâches, courtes et bien définies, doivent être exécutées à maintes reprises et à différents endroits du programme. Au moment d'écrire un tel programme, vous reconnaîtrez ces tâches, et voudrez disposer d'un moyen pratique et économique de les exécuter chaque fois que vous en aurez besoin. Le BASIC vous permet d'isoler de telles tâches à l'intérieur d'éléments appelés *sous-programmes.* Une fois que vous aurez écrit un sous-programme, vous pourrez "l'appeler" à partir de n'importe quel endroit du programme. Il effectuera son travail, puis rendra le contrôle au programme, à l'endroit d'où "l'appel" avait été fait.

Regardons un exemple. Souvenez-vous du petit programme que nous avions examiné, qui lisait les coordonnées d'une adresse (horizontale, verticale) au clavier, et utilisait cette adresse pour placer un pixel sur l'écran. Ce programme deviendra le sous-programme d'un programme plus important que nous écrirons bientôt. Voici à quoi il ressemble, transformé en sous-programme :

```
200 PRINT AT 21,0; "HORIZONTAL"
210 INPUT H
220 PRINT AT 21,0; "VERTICAL"
230 INPUT V
240 PLOT H,V
250 RETURN
```

Vous remarquez qu'une ligne a été ajoutée : une instruction RETURN à la ligne 250. RETURN est un mot clé au-dessus de la touche Y. Chaque sous-programme doit avoir un RETURN; cette instruction indique simplement que le travail du sous-programme est terminé. Quand l'ordinateur rencontre l'instruction RETURN, il

rend automatiquement le contrôle au programme à l'endroit d'où le sous-programme avait été appelé.

Pour appeler le sous-programme, vous pouvez tout simplement introduire la commande :

```
GOSUB 200
```

Nous verrons deux petits programmes utilisant ce sous-programme. Vous introduirez ces deux programmes dans votre ordinateur, ainsi que les lignes du sous-programme, et vous les testerez :

Voici le premier :

```
10 PRINT "TU PEUX METTRE 5 PIXELS"
20 FOR I = 1 TO 5
30 GOSUB 200
40 NEXT I
50 STOP
```

Ce programme vous permet de placer 5 pixels à l'écran. Il commence par écrire un message en haut de l'écran (ligne 10), puis déclenche une boucle FOR. L'instruction GOSUB, à la ligne 30, fait partie de la boucle. Voici la suite d'événements provoqués par cette boucle :

1. L'instruction FOR utilise la variable de contrôle I pour déterminer le nombre de répétitions.
2. L'instruction GOSUB donne le contrôle du programme à la ligne 200, à partir de laquelle le travail d'introduction d'une adresse et d'écriture d'un pixel est fait.
3. La dernière ligne du sous-programme (250) rend le contrôle du programme à la ligne 50, où se trouve l'instruction NEXT.
4. NEXT incrémente la variable de contrôle I, et on continue la boucle.

Vous pouvez voir, par cette suite d'événements, que le sous-programme de la ligne 200 a été appelé cinq fois, ce qui a permis de placer cinq pixels sur l'écran; puis le programme s'est terminé. La ligne 50 est très importante :

```
50 STOP
```

Le mot clé STOP (un caractère majuscule, situé sur la touche A) sert à dire à l'ordinateur que le programme est fini. Dans notre cas, le corps principal du programme est formé des lignes 10 à 50. Quand la boucle FOR a terminé ses cinq répétitions, vous ne voulez pas que l'ordinateur continue jusqu'à la ligne 200, pour exécuter à nouveau le sous-programme. Aussi vous devez dire explicitement à l'ordinateur de s'arrêter, ce qui est le rôle de la ligne 50. C'est la première fois que nous avons eu à utiliser le mot clé STOP. Jusqu'à maintenant, nous avions vu des programmes s'exécutant ligne par ligne de la première à la dernière. Quand l'ordinateur exécute la dernière ligne, il s'arrête automatiquement. Mais dans ce programme, les six dernières lignes (les lignes 200 à 250) constituent un sous-programme, que nous ne voulons exécuter que par appels grâce à l'instruction GOSUB.

Voici le deuxième exemple de programme utilisant notre sous-programme :

```
10 PRINT "PLACE AUTANT DE PIXELS QUE TU VEUX"
20 GOSUB 200
30 GOTO 20
```

Ce programme vous permet de placer sur l'écran autant de pixels que vous voulez. Les lignes 20 et 30 constituent une boucle sans fin. Voici comment travaille le programme :

1. La ligne 20 appelle le sous-programme.
2. Le sous-programme fait son travail, puis rend le contrôle à la ligne 30.
3. L'instruction GOTO de la ligne 30 renvoie le contrôle à la ligne 20.

Exécutez le programme pour voir exactement ce qu'il fait. Vous pouvez continuez à entrer des adresses de pixels aussi longtemps que vous le souhaitez. Après avoir lu chaque adresse, l'ordinateur place un pixel à l'écran. Pour arrêter le programme, vous devez introduire le mot clé STOP au lieu de fournir une adresse. La touche BREAK ne fonctionne pas quand l'ordinateur est en attente de donnée depuis le clavier.

Nous avons un dernier thème à aborder avant de passer aux deux principaux programmes de ce chapitre. Dans le paragraphe

suivant, nous allons voir comment utiliser le mot clé IF pour faire prendre à l'ordinateur des décisions pendant l'exécution d'un programme.

Décisions, décisions...

Vous pouvez souvent accroître la puissance et l'utilité d'un programme en y mettant des instructions permettant à l'ordinateur de "décider" lequel choisir entre deux déroulements différents de l'action. Vous avez peut-être déjà rencontré un problème dans le programme d'écriture de pixels qui aurait pu être résolu en ajoutant au programme quelques lignes décisionnelles. Nous examinerons d'abord le problème, puis nous en verrons la solution.

Exécutez à nouveau le deuxième programme de placements de pixels. Vous savez que lorsque vous voyez le message :

HORIZONTAL

vous devez introduire un nombre inférieur ou égal à 63. De même quand vous voyez le message :

VERTICAL

vous ne pouvez introduire un nombre supérieur à 43. Cela vient de ce que la plus grande adresse que l'instruction PLOT puisse manipuler est 63,43. Toute adresse plus grande tomberait en dehors de l'écran. Peut-être avez-vous déjà vu ce qui arrive au programme si vous entrez accidentellement une adresse " illégale ". Si ce n'est pas le cas, alors essayez maintenant. Introduisez le nombre 64 comme coordonnée horizontale de l'adresse et 44 comme coordonnée verticale. Ces deux nombres sont tous les deux plus grands que ceux que l'instruction PLOT peut accepter. Le résultat est qu'aucun pixel n'est placé sur l'écran; l'exécution du programme est interrompue et le message d'erreur :

B/240

apparaît dans le coin en bas à gauche de l'écran.

Si vous regardez à l'Appendice B, vous trouverez la signification de ce message d'erreur. Le code d'erreur B signifie que le nombre que vous avez introduit ne fait pas partie de l'intervalle de nombres acceptable. Dans notre cas, cet intervalle est défini par l'ordinateur pour le mot clé PLOT. Remarquez que la seconde partie du message d'erreur vous donne le numéro de ligne – 240 – à laquelle l'exécution du programme s'est interrompue. Si vous regardez à nouveau le sous-programme, vous verrez, à coup sûr, que la ligne 240 est celle qui contient l'instruction PLOT.

Pour éviter ce problème, il serait très commode que l'ordinateur vérifie les valeurs introduites pour les variables H et V avant d'exécuter l'instruction PLOT. Si l'une ou l'autre des variables H ou V était détectée comme contenant une valeur trop grande pour être traitée par PLOT, la meilleure chose à faire serait de revenir aux instructions INPUT dans l'attente d'une valeur acceptable. C'est exactement ce que nous ferons faire à l'ordinateur, en utilisant deux instructions IF.

Voici une description générale de l'instruction IF :

IF (expression vraie ou fausse) THEN (instruction mot clé)

Comme vous avez pu le voir, l'instruction IF comporte deux parties. La première commence par le mot clé IF (situé au-dessus de la touche U) et la deuxième par le mot clé THEN (un caractère majuscule, situé sur la touche 3). Après IF, vient une expression que l'ordinateur évalue comme vraie ou fausse. Cette expression possède habituellement la forme d'une égalité ou d'une inégalité. Nous verrons dans un moment comment écrire de telles expressions. Le mot clé THEN est suivi d'une commande à l'ordinateur, mise sous la forme d'une instruction par mot clé.

L'instruction IF a pour résultat l'une des deux alternatives :

1. Si l'expression vraie ou fausse est vraie, alors l'ordinateur exécute l'instruction placée après THEN.
2. Si l'expression vraie ou fausse est fausse, alors l'ordinateur saute l'instruction placée après THEN, et passe tout simplement à la ligne suivante du programme.

Vous pouvez voir à quel point l'instruction IF est puissante. De cette façon, vos programmes peuvent être conçus pour réagir

correctement et efficacement à de nombreuses situations différentes possibles lors de leur exécution.

Voyons précisément comment écrire une instruction IF. L'expression vraie ou fausse nous amène à apprendre à nous servir d'un nouvel ensemble de symboles du clavier. Ce sont les symboles d'égalité et d'inégalité :

```
=    (est égal à)
<    (est inférieur à)
>    (est supérieur à)
<=   (est inférieur ou égal à)
>=   (est supérieur ou égal à)
<>   (est différent de)
```

Tous ces symboles sont des caractères majuscules, situés respectivement sur les touches L, N, M, R, Y et T. Ces symboles sont utilisés pour comparer deux valeurs pouvant être numériques ou non-numériques. Voici des exemples d'expressions utilisant ces symboles :

```
I > 5
K < J
M <= 16
S$ = "OUI"
H < > 15
```

Tous les exemples donnés ci-dessus comparent les valeurs de deux variables différentes, ou la valeur d'une variable avec un littéral. Chacune de ces expressions est, soit vraie, soit fausse. Par exemple, dans la première de ces expressions, si la variable I contient la valeur 4, l'expression est fausse, car 4 n'est pas supérieur à 5. Cependant, si I contient la valeur 7, l'expression est vraie, car 7 est supérieur à 5.

Voici quelques exemples d'instructions IF utilisant des expressions vraies ou fausses :

```
IF I > 5 THEN PRINT "I EST TROP GRAND"
IF K > J THEN INPUT J
IF M <= 16 THEN GOTO 200
IF A$ = "OUI" THEN STOP
IF H < > 15 THEN LET H = 100
```

Remarquez que dans chacun des cas, THEN est suivi d'un mot clé. Tous les mots clés de votre clavier peuvent être utilisés après THEN, dans une instruction IF, bien que certains soient utilisés plus fréquemment que d'autres. Gardez à l'esprit que l'instruction qui suit le THEN n'est exécutée *que si* l'expression vraie ou fausse est évaluée à *vraie*. Autrement, l'ordinateur passe tout simplement à la ligne suivante du programme.

Maintenant, comme exemples significatifs d'instructions IF, revenons au problème que nous avons esquissé plus haut. La solution du problème réside en un perfectionnement du sous-programme commençant à la ligne 200. Quand l'ordinateur lit chaque valeur depuis le clavier, et affecte respectivement ces valeurs aux variables H et V, nous aimerions ajouter un *test* pour nous assurer que les deux valeurs sont dans l'intervalle acceptable par l'instruction PLOT. Nous pouvons aisément énoncer ce test sous la forme d'une instruction IF située juste après chacun des ordres INPUT.

Voici la première :

```
210 INPUT H
215 IF H > 63 THEN GOTO 210
```

La ligne 210 sert à lire une valeur H. La ligne 215 sert à évaluer l'expression H < 63 afin de déterminer quelle action entreprendre : si la variable H contient une valeur supérieure à 63, alors c'est l'instruction GOTO qui est exécutée. Le contrôle du programme revient à la ligne 210, pour l'introduction d'une nouvelle valeur. Sinon, dans le cas où la valeur H est inférieure ou égale à 63, l'instruction GOTO est sautée, et le programme passe normalement à l'instruction suivante.

Nous pouvons écrire une instruction IF similaire juste après l'instruction INPUT de V. Cette deuxième instruction IF, à la ligne 235, évalue l'expression "V > 43"; elle rend le contrôle à la ligne 230, si cette dernière expression est vraie.

Le sous-programme complet apparaît à la Figure 3.7. Etudiez-le soigneusement, et assurez-vous de bien avoir compris le fonctionnement des instructions IF. Puis expérimentez ce fonctionnement avec l'un des deux petits programmes que nous avons décrits plus haut. En particulier, quand le programme fonctionne, essayez d'introduire des valeurs incorrectes pour les coordonnées hori-

zontales et verticales des adresses. Maintenant, au lieu de provoquer la fin du programme, ce genre d'erreur de saisie entraînera simplement le rejet de la valeur introduite. Par exemple, en voyant le message :

HORIZONTAL

essayez d'introduire le nombre 66. Quand vous le faites, le message reste identique, ce qui indique que 66 ne convient pas pour une adresse horizontale. Vous devez introduire une autre valeur.

Comme rappel de tout ce que vous avez appris jusqu'à présent dans ce chapitre, nous examinerons maintenant un programme qui trace des rectangles à l'écran. Nous l'appellerons le programme de rectangles.

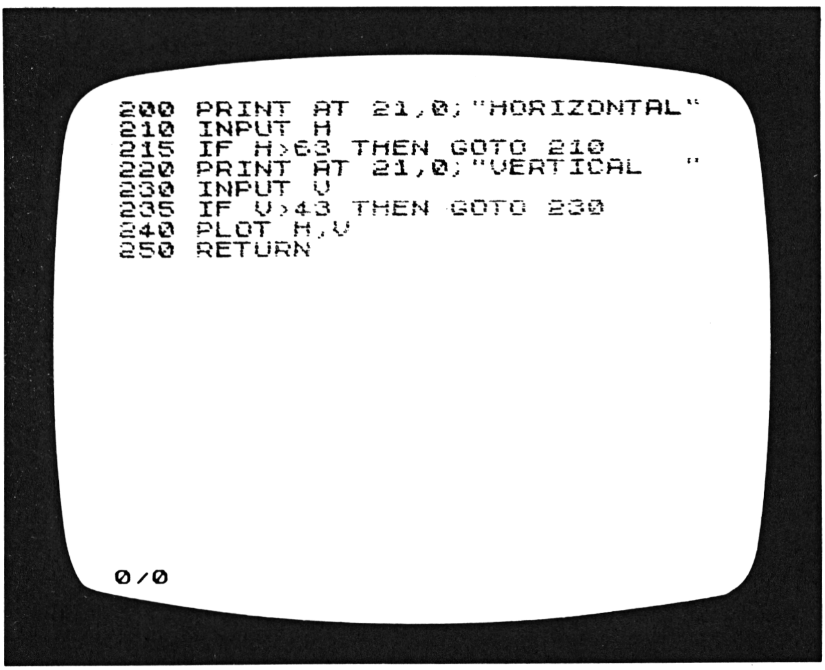


Figure 3.7 : Sous-programme Pixel contenant des instructions IF

Le programme de rectangles

Le corps principal du programme est présenté à la Figure 3.8. Le programme utilise le sous-programme de la Figure 3.7, aussi vous aurez à y insérer ces lignes lorsque vous taperez le programme. Introduisez maintenant le programme complet dans votre ordinateur et exécutez-le.

Ce programme est simplement un exercice conçu pour montrer en action un certain nombre d'outils de programmation. De plus, vous pouvez l'utiliser pour produire quelques images intéressantes sur l'écran. Voici comment il fonctionne : il commence par afficher la demande de deux adresses complètes de pixels. En d'autres termes, vous verrez les messages HORIZONTAL et VERTICAL deux fois, l'ordinateur attendant après chaque message que vous ayez introduit un nombre. Un pixel apparaîtra à l'écran

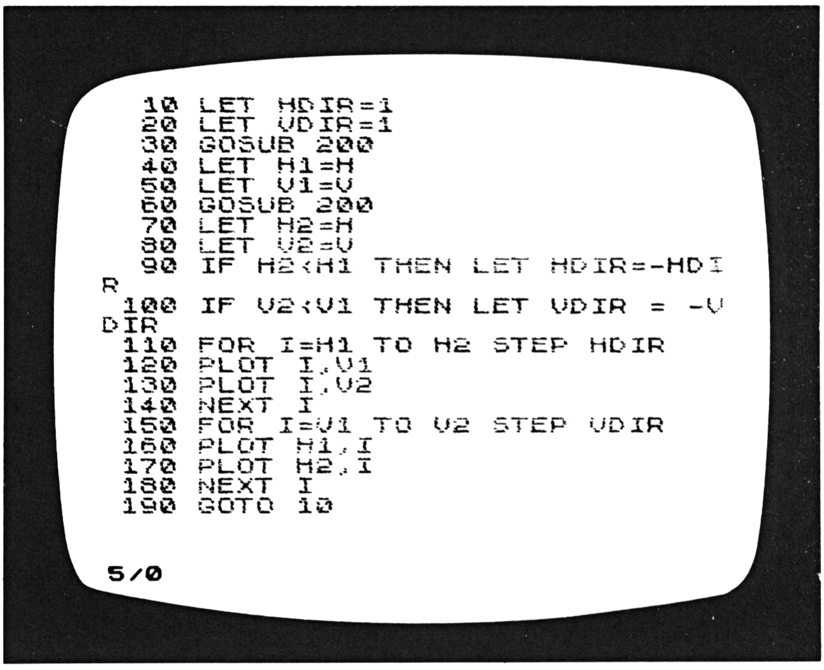


Figure 3.8 : Programme Rectangle

dès que vous aurez introduit une adresse complète et correcte. Quand les deux pixels auront été définis, l'ordinateur tracera un rectangle à l'écran, en utilisant les pixels en tant que coins opposés diagonalement du rectangle. Ce processus continuera aussi longtemps que vous le voudrez, vous permettant de tracer à l'écran de nombreux rectangles de tailles et formes différentes. La Figure 3.9 montre un exemple de figures produites par ce programme. Pour terminer le programme, introduisez le mot clé STOP.

L'expérimentation du programme vous permettra de remarquer qu'il offre la possibilité de tracer des rectangles dans des directions différentes. Pour voir ce que cela signifie, commencez par introduire les coordonnées 5,5 et 25,25. Puisque la première coordonnée est plus basse, et plus à gauche que la seconde, l'ordinateur trace le haut et le bas du rectangle de gauche à droite, et les côtés de bas en haut. Maintenant introduisez deux

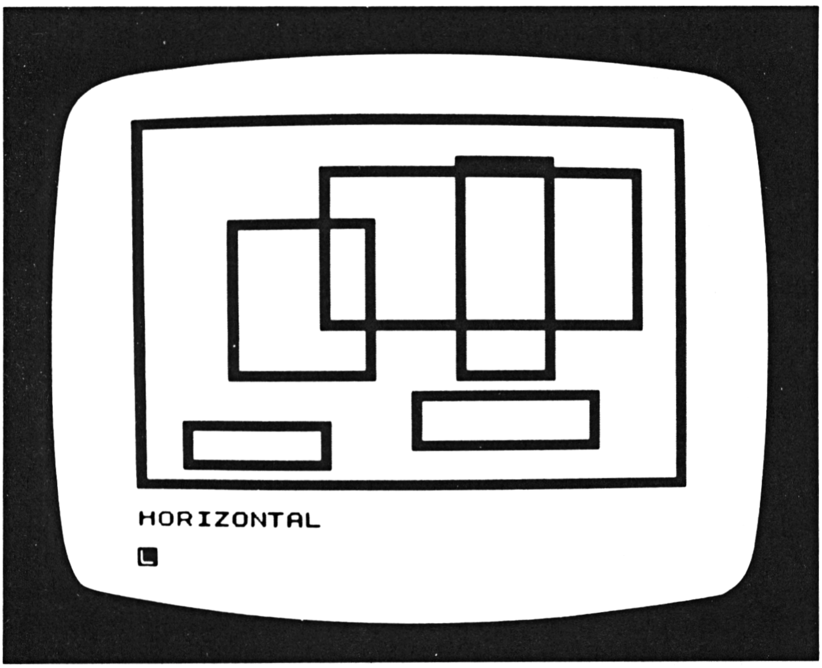


Figure 3.9 : Résultats du programme Rectangle

coordonnées : 45,43 et 25,23, dans cet ordre. Dans ce cas, la première coordonnée introduite se trouve au-dessus et à droite de la seconde, et il en résulte que le rectangle est tracé dans des directions opposées à celles du premier rectangle. Le haut et le bas sont tracés de droite à gauche; et les côtés de haut en bas. Ceci peut paraître un détail négligeable, mais c'est une caractéristique du programme qui a été soigneusement prévue. Examinons les lignes du programme. Pour regarder la liste sur votre écran, entrez STOP pour terminer l'exécution du programme, puis appuyez à nouveau sur NEW LINE (ou ENTER). Le programme est trop long pour être affiché en entier à l'écran, mais souvenez-vous qu'il est possible d'utiliser le mot clé LIST pour visualiser les différentes parties du programme. Par exemple, si vous voulez voir le sous-programme de la ligne 200, vous n'avez qu'à taper :

```
LIST 200
```

Les deux premières lignes du programme définissent les variables HDIR et VDIR :

```
10 LET HDIR = 1
20 LET VDIR = 1
```

Les noms de ces variables représentent respectivement les directions *horizontale* et *verticale.* Durant le déroulement du programme, chacune de ces variables contiendra toujours l'une des deux valeurs 1 ou −1; elles seront utilisées pour déterminer dans quelle direction tracer le rectangle. Nous verrons comment plus loin en étudiant les lignes du programme.

Les lignes 30 et 60 comportent toutes les deux un appel au sous-programme de la ligne 200 :

```
GOSUB 200
```

Souvenez-vous de ce que fait ce sous-programme :

1. Il affiche le message HORIZONTAL à l'écran, puis lit une valeur (inférieure ou égale à 63) depuis le clavier. Il range cette valeur dans la variable H.
2. Il affiche le message VERTICAL à l'écran, puis lit une valeur (inférieure ou égale à 43) depuis le clavier. Il range cette valeur dans la variable V.

3. Il utilise l'instruction PLOT pour placer un pixel à la coordonnée H,V.

Ce sous-programme doit être appelé deux fois pour chaque rectangle – pour déterminer les deux coins diagonalement opposés qui définissent le rectangle. Chaque fois que le sous-programme est appelé, il rangera de nouvelles valeurs dans les variables H et V. Les anciennes valeurs de H et V seront perdues. Mais dans le but de tracer le rectangle, le programme doit garder en mémoire les coordonnées respectives des deux coins. C'est l'objet des lignes 40, 50, 70 et 80 de ce programme. Après chaque appel du sous-programme, le programme stocke les nouvelles valeurs de H et V dans des variables dont le but est la *sauvegarde* de ces valeurs. Pour être précis, on sauvegarde les valeurs de la première coordonnée (c'est-à-dire celle du premier appel au sous-programme) dans H1 et V1, et les valeurs de la deuxième coordonnée (c'est-à-dire celle du deuxième appel au sous-programme) dans H2 et V2.

Etudions la forme des instructions LET affectant des valeurs aux variables H1, V1, H2, et V2. Par exemple, examinons la première :

```
40 LET H1 = H
```

Cette instruction, d'une façon différente, dit : "Prends la valeur actuelle de la variable H, et range cette valeur dans la variable H1". C'est la première fois que nous voyons un nom de variable à droite du signe égal d'une instruction LET. Il est important que vous compreniez ce que cette instruction fait – et ce qu'elle *ne fait pas.* Elle définit la variable H1 et lui affecte une valeur. Mais elle *ne modifie en rien* la variable H. Une fois que cette instruction LET a été exécutée, la variable H conserve la valeur qu'elle avait auparavant.

Une fois que ces deux coordonnées ont été déterminées, l'ordinateur doit décider dans quelles directions tracer le rectangle. Les lignes 90 et 100 prennent cette décision. Souvenez-vous que HDIR et VDIR ont toutes deux été initialisées à la valeur 1. Comme nous le verrons, cette valeur indique que le rectangle sera tracé de gauche à droite et de bas en haut. Le but des lignes 90 et 100 est alors de *modifier* ces directions, si cela est nécessaire. Si la deuxième valeur horizontale H2 est inférieure à la

première, H1 (c'est-à-dire à gauche de celle-ci), le rectangle doit être tracé de droite à gauche. Ceci est indiqué en changeant la valeur de HDIR de 1 en −1 :

```
90 IF H2 < H1 THEN LET HDIR =− HDIR
```

Remarquez l'instruction LET qui se trouve après le mot THEN. La variable HDIR apparaît *à la fois* des deux côtés du signe égal. Cette instruction LET a pour signification : "Prends l'*inverse* de la valeur actuellement rangée dans HDIR, et remets cette nouvelle valeur dans HDIR." Autrement dit, puisque la valeur initiale de HDIR était égale à 1, la nouvelle valeur sera égale à −1. La valeur initiale est bien sûr perdue.

Ensuite dans le programme on a deux boucles FOR. La première boucle (les lignes 110 à 140), trace les lignes du haut et du bas du rectangle, et la seconde boucle (les lignes 150 à 180), trace les côtés. En étudiant la façon dont ces boucles sont construites, vous comprendrez pleinement le sens des variables de direction, HDIR et VDIR.

Chaque boucle contient deux instructions PLOT. Nous avons déjà vu comment deux instructions PLOT distinctes, à l'intérieur d'une boucle FOR, pouvaient produire deux lignes distinctes. Dans notre cas, elles produisent deux lignes parallèles. Dans la première boucle, la variable de contrôle I est incrémentée de la valeur de H1 à la valeur de H2 :

```
110 FOR I = H1 TO H2 STEP HDIR
```

Les instructions PLOT qui suivent ont leurs coordonnées verticales égales respectivement à V1 et V2; elles utilisent la valeur de I qui évolue comme coordonnées horizontales :

```
120 PLOT I,V1
130 PLOT I,V2
```

C'est pour cela que vous voyez les lignes du haut et du bas de chaque rectangle se tracer d'abord quand le programme fonctionne.

De la même façon, la seconde boucle FOR trace les côtés du rectangle en maintenant constantes les coordonnées horizontales,

H1 et H2, et en faisant varier les coordonnées verticales, de V1 à V2 :

```
150 FOR I = V1 TO V2 STEP VDIR
160 PLOT H1,I
170 PLOT H2,I
180 NEXT I
```

Maintenant, que dire de la clause STEP de ces deux instructions FOR ? Le moyen le plus simple de voir en quoi elle est nécessaire est de penser à l'introduction des coordonnées de deux pixels, telles que le premier soit au-dessus et à droite du second; par exemple :

```
coordonnée P1 : 45,43
coordonnée P2 : 25,23
```

Substituez maintenant ces valeurs à l'intérieur des instructions FOR, et voyez ce que vous aurez :

```
110 FOR I = 45 TO 25
150 FOR I = 43 TO 23
```

Il apparaît que ces deux boucles FOR, telles que données ci-dessus, n'exécuteront même pas une seule fois leurs instructions; en effet, le premier nombre de l'instruction FOR est plus grand que le second. Pour permettre à ces boucles de fonctionner, nous devons ajouter une clause STEP qui aura pour effet de faire *diminuer* (ou de *décrémenter*) la variable de contrôle I à chaque répétition de la boucle. C'est pourquoi les variables de direction HDIR et VDIR doivent être positionnées à −1 si la première coordonnée est plus grande que la deuxième. Le résultat sera que la boucle FOR pour les coordonnées ci-dessus deviendra :

```
110 FOR I = 45 TO 25 STEP - 1
150 FOR I = 43 TO 23 STEP - 1
```

La clause STEP précise que chaque variable de contrôle sera diminuée de 1 à chaque répétition de la boucle.

Si ces détails vous semblent encore un peu confus, exécutez à nouveau le programme, et réfléchissez aux deux boucles FOR tout

en regardant apparaître les rectangles à l'écran. Dessinez plusieurs rectangles et observez les différentes directions dans lesquelles ils sont tracés. Vous savez maintenant que ces directions sont déterminées par l'ordre dans lequel vous introduisez les coordonnées des pixels.

La dernière ligne du corps principal de ce programme (c'est-à-dire la partie qui précède le sous-programme) est à la ligne 190 :

```
190 GOTO 10
```

Cette ligne crée une boucle sans fin. Vous pouvez continuer à tracer des rectangles aussi longtemps que vous le voulez. Le programme lui-même ne contient pas de clause d'arrêt, et c'est pourquoi vous devrez utiliser la commande STOP afin de terminer l'exécution du programme.

Le dernier programme de ce chapitre est également un programme graphique, mais qui est certainement plus intéressant que le programme de rectangles. Nous appellerons ce dernier programme le "programme de dessin", car il a été conçu pour vous permettre de tracer des dessins sur l'écran. Nous commencerons par vous donner le mode d'emploi du programme, afin que vous puissiez le taper et commencer à vous amuser immédiatement. (Le programme est long, aussi pensez à bien le copier sur cassette dès que vous aurez fini de le saisir.)

Vous trouverez également une explication, ligne par ligne, du fonctionnement de ce programme. Ce programme a été inséré dans ce chapitre dans le but d'approfondir et de développer votre compréhension de toutes les instructions BASIC que vous avez pu apprendre jusqu'ici. Les méthodes et les techniques présentées dans le "programme de dessin" sont plus perfectionnées que tout ce que vous avez vu jusqu'à présent. Vous pensez peut-être qu'il vous faut un certain temps avant de comprendre complètement comment il fonctionne. Mais ne vous précipitez pas. Vous pouvez prendre plaisir à *utiliser* le programme, même si vous n'avez pas complètement brisé la coquille de sa logique interne.

LE PROGRAMME DE DESSIN

La liste du programme est présentée aux Figures 3.10, 3.11 et 3.12. Comme vous pouvez le voir, le programme est trop long pour pouvoir être affiché en une seule fois à l'écran. En introduisant le programme dans l'ordinateur, et en remplissant l'écran de lignes d'instructions, vous verrez que l'ordinateur imprime et réimprime sans cesse la liste du programme à l'écran. Il ajuste toujours la partie du programme apparaissant à l'écran, de façon que la ligne actuelle soit affichée. Ce processus d'ajustement peut être assez lent lorsque vous introduisez un long programme comme celui-ci. Pour accélérer le processus, vous pouvez mettre l'ordinateur en mode rapide. Introduisez simplement le mot clé FAST en tant que commande en mode immédiat. Si vous le faites, vous verrez que l'ordinateur met

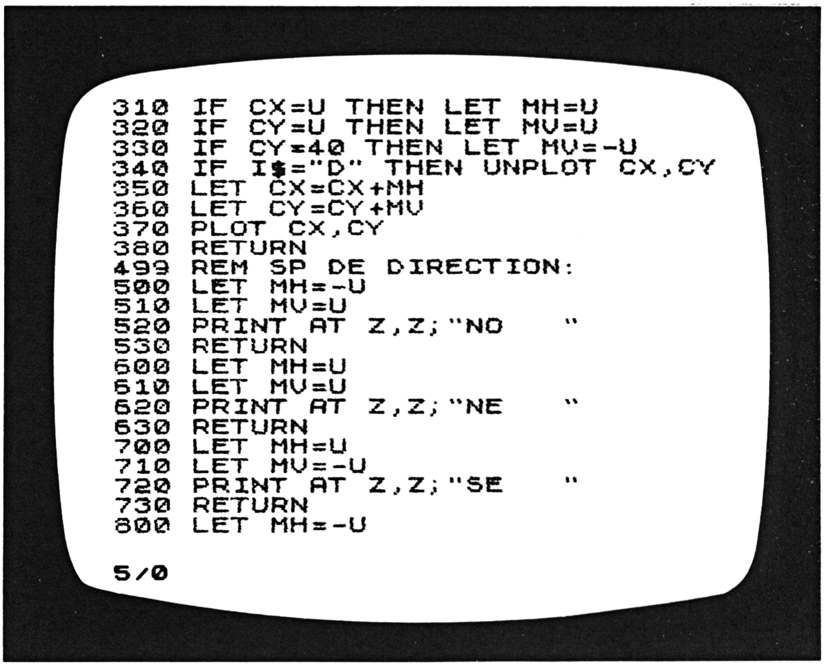
```
  10 CLS
  20 PRINT TAB 5;"D=DEPL,I=IMPRI
M,E=EFF"
  30 LET U=1
  40 LET Z=0
  50 LET CX=31
  60 LET CY=21
  70 REM PREMIER PIXEL
  80 GOSUB 1100
  90 PLOT CX,CY
 100 REM LECT CLAVIER
 110 IF INKEY$="" THEN GOTO 110
 120 LET I$=INKEY$
 130 IF I$="E" THEN GOTO 10
 140 REM DEPL/IMPR
 150 IF I$="I" OR I$="D" THEN GO
SUB 300
 160 IF I$>="1" AND I$<="8" THEN
 GOSUB (100*(VAL (I$)+4))
 170 GOTO 110
 299 REM IMPR OU DEPL
 300 IF CX=63 THEN LET MH=-U

5/0
```

Figure 3.10 : Programme de dessin

moins de temps pour ajuster la liste du programme. Une fois que vous aurez introduit le programme, et avant de l'exécuter, remettez l'ordinateur en mode lent en introduisant le mot clé SLOW, toujours en tant que commande en mode immédiat.

Vérifiez toutes les lignes du programme afin de vous assurer que vous l'avez introduit correctement. Puis introduisez la commande RUN pour exécuter le programme. La Figure 3.13 montre l'écran tel qu'il apparaît au début, lorsque le programme démarre pour la première fois. Au centre de l'écran, se trouve un unique pixel, et quelques informations brèves sont affichées en haut de l'écran. Vous pouvez déplacer ce pixel n'importe où sur l'écran, dans l'une quelconque des huit directions : vers le haut, vers le bas, à gauche ou à droite; ainsi qu'en diagonale : Nord-Ouest, Nord-Est, Sud-Est, Sud-Ouest. Le mot se trouvant dans le coin en haut à gauche de l'écran indique toujours la direction actuelle. (Au début du programme, la direction est "vers



```
310 IF CX=U THEN LET MH=U
320 IF CY=U THEN LET MV=U
330 IF CY=40 THEN LET MV=-U
340 IF I$="D" THEN UNPLOT CX,CY
350 LET CX=CX+MH
360 LET CY=CY+MV
370 PLOT CX,CY
380 RETURN
499 REM SP DE DIRECTION:
500 LET MH=-U
510 LET MV=U
520 PRINT AT Z,Z;"NO     "
530 RETURN
600 LET MH=U
610 LET MV=U
620 PRINT AT Z,Z;"NE     "
630 RETURN
700 LET MH=U
710 LET MV=-U
720 PRINT AT Z,Z;"SE     "
730 RETURN
800 LET MH=-U

5/0
```

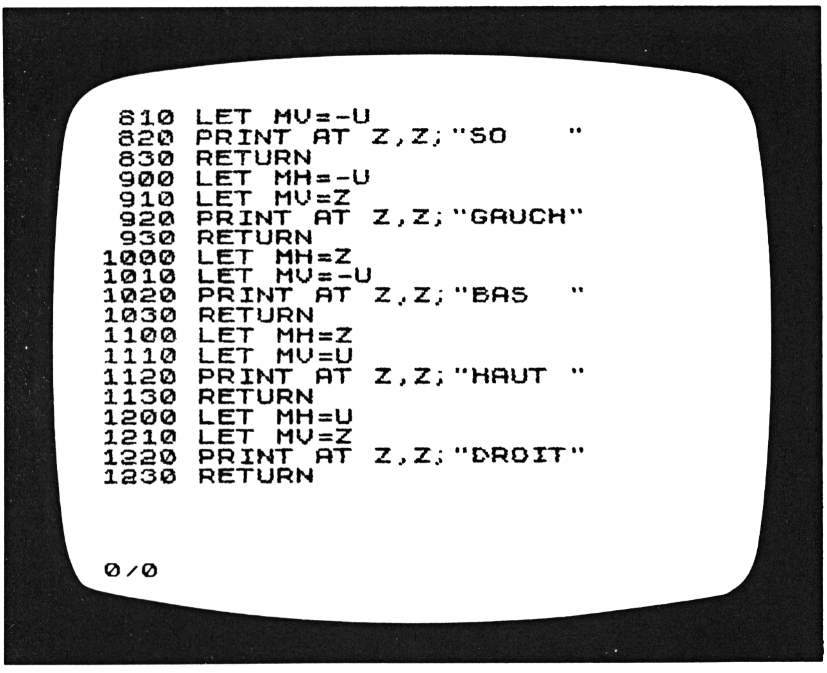
Figure 3.11 : Programme de dessin (suite)

le haut"). Vous pouvez modifier la direction en appuyant sur l'une des 8 touches de nombres, de 1 à 8. Pour les directions en diagonale, appuyez sur les touches 1 à 4. Vous pouvez lire les directions sur les caractères graphiques de ces touches :

1. Nord-Ouest
2. Nord-Est
3. Sud-Est
4. Sud-Ouest

Les autres directions sont indiquées par les flèches que l'on trouve sur les touches 5 à 8 :

5. à gauche
6. vers le bas
7. vers le haut
8. à droite



```
 810 LET MV=-U
 820 PRINT AT Z,Z;"SO    "
 830 RETURN
 900 LET MH=-U
 910 LET MV=Z
 920 PRINT AT Z,Z;"GAUCH"
 930 RETURN
1000 LET MH=Z
1010 LET MV=-U
1020 PRINT AT Z,Z;"BAS   "
1030 RETURN
1100 LET MH=Z
1110 LET MV=U
1120 PRINT AT Z,Z;"HAUT  "
1130 RETURN
1200 LET MH=U
1210 LET MV=Z
1220 PRINT AT Z,Z;"DROIT"
1230 RETURN

0/0
```

Figure 3.12 : Programme de dessin (suite)

Essayez d'appuyer sur ces huit touches, une seule à la fois. A chaque fois que vous appuierez sur une touche, le mot du coin en haut à gauche de l'écran changera.

Une fois que vous avez sélectionné la direction dans laquelle vous voulez déplacer le pixel, vous pouvez le faire en appuyant sur l'une des deux touches : la touche D (comme Déplacer) déplace tout simplement le pixel. Appuyez sur la touche une seule fois pour le déplacer d'une position dans la direction sélectionnée, ou maintenez votre doigt sur la touche D pour obtenir un déplacement continu.

La touche I (comme Imprimer) laisse une trace de pixels derrière celui qui se déplace. Une nouvelle fois, vous pouvez soit appuyer sur la touche pour un seul mouvement, soit tenir la touche enfoncée pour plusieurs déplacements. Prenez un moment pour essayer ces deux touches, maintenant dans plusieurs directions.

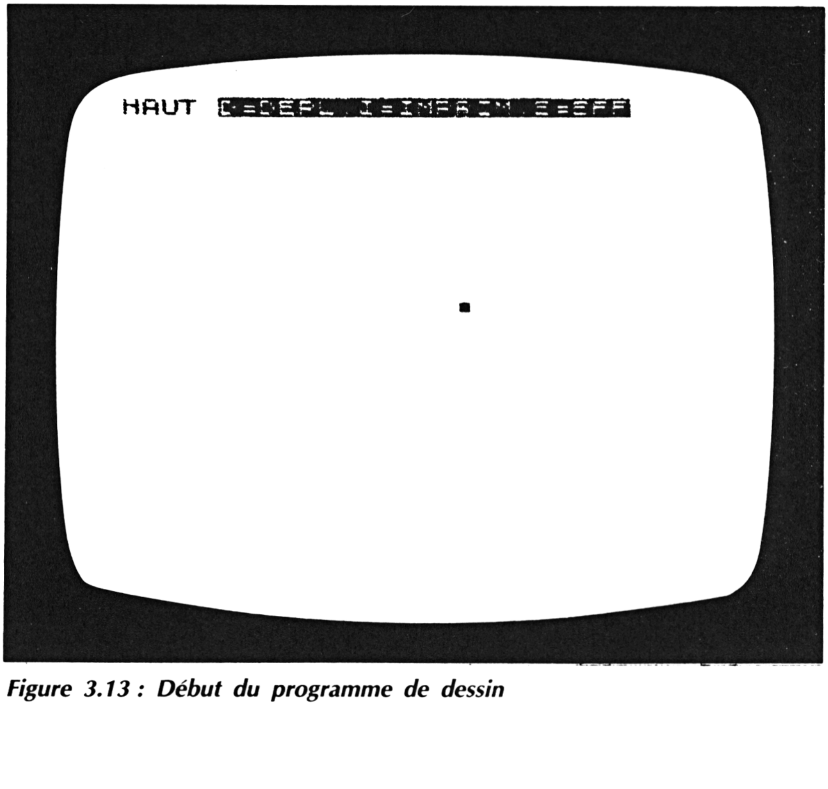


Figure 3.13 : Début du programme de dessin

Vous pouvez effacer n'importe quel pixel de l'écran en repassant sur lui en arrière. Appuyez sur la touche de direction correcte, vers le pixel que vous voulez effacer, puis appuyez sur la touche D.

Si vous désirez repartir avec un écran vide, appuyez sur la touche E (comme Effacer), pour "effacer" le graphisme actuel.

Quand le déplacement du pixel atteint l'un des bords de l'écran, il est automatiquement "renvoyé" vers l'écran. Si le déplacement se fait en diagonale, l'angle de réflexion est de 90 degrés. Si c'est vers le haut, le bas, la gauche ou la droite, la réflexion se fait dans la direction opposée.

Vous verrez que plus longtemps vous jouerez avec ce programme, meilleurs seront vos résultats. Vous pouvez l'utiliser pour tracer pratiquement n'importe quoi à l'écran. La principale limitation, bien sûr, est celle de la résolution déterminée par la taille d'un pixel. La Figure 3.14 présente un exemple de dessin tracé en utilisant ce programme.

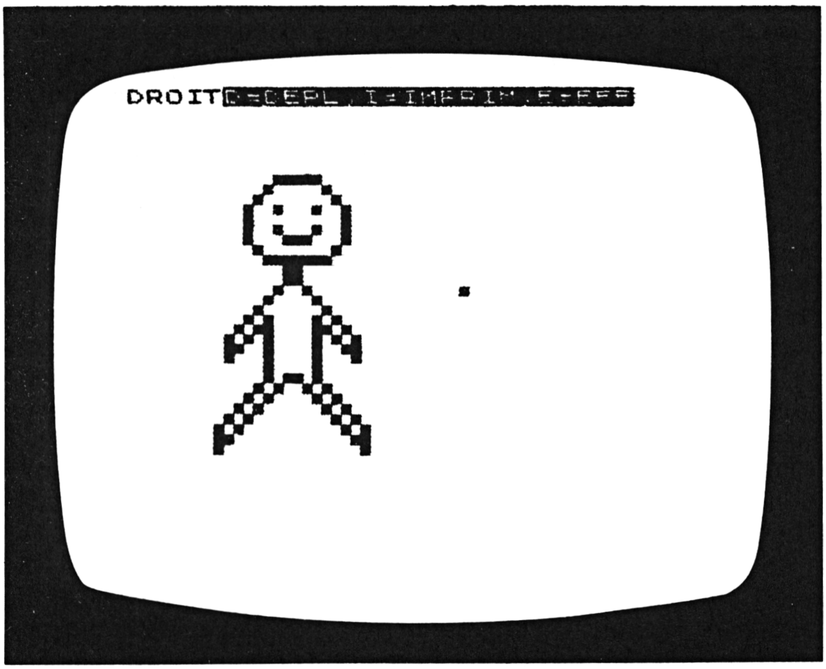


Figure 3.14 : Exemple de dessin

A l'intérieur du programme

Appuyez sur la touche BREAK pour interrompre le programme; puis appuyez sur NEW LINE (ou ENTER) pour lister le programme à l'écran. Ce programme est organisé en une partie principale de contrôle, et un certain nombre de sous-programmes. Nous verrons d'abord comment fonctionne la partie principale du programme en liaison avec les sous-programmes; puis nous regarderons en détail un certain nombre de lignes du programme.

Ce programme met en évidence l'énorme puissance des sous-programmes. Les principales tâches de ce programme sont structurées en sous-programmes spécialisés. Cela donne un programme facile à comprendre et facile à corriger ou à modifier. Une fois que vous aurez compris le rôle de chaque sous-programme, vous saurez exactement où regarder si vous voulez modifier un détail quelconque du programme.

En bref, voici comment est organisé le programme. Huit petits sous-programmes, en fin de programme, sont chargés des tâches de changement de la direction du mouvement. Chaque fois que vous appuyez sur l'une des touches de direction, l'un de ces sous-programmes est appelé. Ces sous-programmes déterminent les valeurs de deux variables, qui se nomment MH (comme Mouvement Horizontal) et MV (comme Mouvement Vertical). Ces deux variables déterminent, tour à tour, la direction dans laquelle le pixel se déplacera quand vous appuierez sur les touches I ou D. Les variables MH et MV ont toujours l'une des trois valeurs : 1, 0 ou −1. En plus d'affecter des valeurs à ces variables, les sous-programmes de direction affichent la direction actuelle dans le coin en haut à gauche de l'écran. Chacun de ces sous-programmes ne fait que quatre lignes de long, et chacun commence à un numéro de ligne qui est un multiple de 100 (500, 600, 700...). C'est quelque chose de très important pour l'instruction GOSUB qui appelle ces sous-programmes, comme nous le verrons bientôt.

Le sous-programme qui déplace le pixel est situé à la ligne 300. Il est plus long et plus compliqué que les sous-programmes de direction. D'abord il doit vérifier si la position actuelle du pixel n'est pas sur l'un des quatre bords de l'écran. Si c'est le cas, le sous-programme doit modifier l'une des variables MH ou MV

pour inverser le sens du mouvement. Puis le sous-programme doit déterminer si le pixel se déplace simplement, ou bien s'il laisse une trace derrière lui (c'est-à-dire si vous avez appuyé sur la touche D ou sur la touche I). Si le pixel ne fait que se déplacer, la précédente position sur l'écran doit être remise à blanc (par l'instruction UNPLOT). Enfin, ce sous-programme détermine la nouvelle adresse du pixel, en additionnant MH ou MV aux coordonnées actuelles, et écrit un pixel (par PLOT) à la nouvelle position.

Les lignes 110 à 170 effectuent la lecture depuis le clavier, et appellent le sous-programme adéquat, en fonction de la touche sur laquelle vous avez appuyé. Ces lignes constituent une boucle sans fin, comme vous pouvez le voir à la ligne 170 :

```
170 GOTO 110
```

Il en résulte que l'ordinateur passe beaucoup de temps simplement à attendre que vous appuyiez sur une des touches du clavier. Dans ce programme, contrairement à ceux que vous avez déjà vus jusqu'à maintenant, on n'utilise pas le mot clé INPUT pour lire au clavier. A la place, nous utilisons une fonction programmée nommée INKEY$ (située au-dessous de la touche B de votre clavier). Nous verrons en quoi INKEY$ diffère de INPUT quand nous détaillerons ces lignes.

En définitive, en reprenant l'examen du programme, nous pouvons dire que les lignes 10 à 90 constituent la partie "d'initialisation" de ce dernier. La ligne 10 remet l'écran à blanc (par l'instruction CLS). La ligne 20 affiche la ligne de "menu" du haut de l'écran en vidéo inversée, en utilisant la fonction TAB pour placer le message à 5 espaces du bord gauche de l'écran. Les lignes 30 à 60 définissent (ou "initialisent") plusieurs variables importantes; aussi pourrions-nous commencer notre examen détaillé en décrivant ces variables.

Les lignes 30 et 40 définissent les variables U (comme Unité) et Z (comme Zéro) :

```
30 LET U = 1
40 LET Z = 0
```

L'objet de ces deux variables est une histoire assez longue. Elle concerne davantage la nature de votre ordinateur que ce pro-

gramme en particulier. Comme vous le savez, votre ordinateur possède une limitation en taille mémoire. Ce programme est prévu pour tenir – et fonctionner – dans une mémoire de 2 K. Quand vous commencerez l'écriture de programmes assez longs, comme celui-ci – en particulier de programmes faisant un usage intensif de l'écran TV pendant leur déroulement – vous commencerez également à vous soucier des moyens pour économiser l'espace mémoire. N'oubliez pas que l'ordinateur doit utiliser sa mémoire pour nombre de tâches différentes, quand vous exécutez un programme. Une partie de la mémoire est réservée au rangement des instructions du programme lui-même; une autre sert à garder trace de l'information qui est affichée à l'écran; et d'autres tâches, moins visibles, nécessitent encore de la mémoire. Un moyen évident d'économiser de la mémoire, est d'en utiliser le moins possible pour le programme lui-même. Dans le programme de dessin, voilà pourquoi on a introduit les variables U et Z.

L'ordinateur a besoin de beaucoup moins de mémoire pour ranger un caractère que pour ranger un nombre. L'ordinateur utilise un procédé spécial de stockage des nombres; ce système (appelé *notation flottante,* au cas où vous voudriez aller plus loin) est conçu pour stocker des nombres avec le maximum de précision, dans un espace mémoire relativement réduit. La précision, naturellement, est quelque chose qui prend pour vous de l'importance quand vous commencez à utiliser l'ordinateur pour des calculs. Ce procédé utilise beaucoup plus de mémoire pour chaque nombre littéral qu'il n'en faut pour un caractère. Aussi, chaque fois qu'un nombre apparaît dans la liste de votre programme, vous pouvez vous demander s'il existe un moyen quelconque pour remplacer ce nombre par une lettre.

Dans le programme de dessin, les nombres 0 et 1 sont utilisés de nombreuses fois. Le sachant, nous pouvons imaginer une petite astuce pour gagner un peu d'espace mémoire. Nous pouvons affecter les valeurs 0 et 1 à deux variables au début du programme, puis utiliser ces variables pendant le programme pour symboliser les valeurs 0 et 1. C'est exactement ce que font les lignes 30 et 40. Ainsi lorsque vous lirez ce programme, chaque fois que vous verrez les variables nommées Z et U, les considérerez-vous comme les valeurs "constantes" zéro et un. Ceci peut

rendre le programme un peu plus difficile à déchiffrer, mais le résultat en vaut la peine. Après tout la capacité de votre mémoire est beaucoup plus grande que la capacité de celle de l'ordinateur.

Les deux autres variables définies en début de programme ne demanderont pas une explication aussi longue. Les variables CX et CY représentent respectivement les coordonnées actuelles, horizontale et verticale. (Si vous avez des notions de mathématiques, vous savez que les lettres X et Y sont utilisées traditionnellement comme noms de variables pour représenter dans un graphe les coordonnées horizontale et verticale d'un point.) Les lignes 50 et 60 initialisent ces deux variables :

```
50 LET CX = 31
60 LET CY = 21
```

Ceci veut dire que le pixel que vous voyez, à peu près au centre de l'écran au début de l'exécution du programme, a comme coordonnées 31,21.

La ligne 70 est la première des quelques lignes REM du programme. Le mot clé REM signifie "Commentaire" *(REMark)*. Il est situé au-dessus de la touche E de votre clavier). Les lignes REM ne provoquent aucune action de la part de l'ordinateur; pendant l'exécution du programme, l'ordinateur les ignore. L'utilité de REM réside dans la possibilité que cette instruction vous donne de "documenter" votre programme à l'intérieur de la liste du programme elle-même. Après le mot clé REM, vous pouvez écrire ce que vous voulez. Habituellement, vous n'écrivez que quelques mots courts – les vôtres, pas des mots en BASIC – qui décriront ce qui se passe à un certain endroit du programme. Puis, chaque fois que vous regarderez la liste de votre programme, les lignes REM seront là pour vous aider à vous souvenir de ce que fait le programme.

Malheureusement, les lignes REM prennent de la place dans la mémoire de l'ordinateur. Pour cette raison, vous devez les utiliser modérément. De plus, si en développant un programme vous manquez de mémoire, vous pouvez toujours supprimer (ou condenser) n'importe lesquelles des lignes REM que vous aviez écrites. Mais en général, vous vous apercevrez qu'un commentaire bien écrit et placé au bon endroit, sous forme de ligne REM, peut vous économiser pas mal de temps, quand vous essayez de

modifier, corriger ou tout simplement de comprendre un programme.

La partie initialisation du programme possède deux autres lignes. La ligne 80 initialise la direction du mouvement en appelant l'un des sous-programmes de direction :

```
80 GOSUB 1100
```

Le sous-programme de la ligne 1100 est celui qui positionne la direction "vers le haut".

Enfin, la ligne 90 trace le premier pixel :

```
90 PLOT CX,CY
```

C'est le pixel qui se trouve au centre de l'écran au début de l'exécution du programme.

La partie initialisation du programme est exécutée une seule fois quand le programme commence, puis à nouveau à chaque fois que vous appuyez sur la touche E pour effacer l'écran. La ligne 130 utilise une instruction GOTO pour "ré-initialiser" le programme, si vous entrez le caractère "E" :

```
130 IF I$ = "E" THEN GOTO 10
```

La ligne 110 lit le clavier, en utilisant l'instruction INKEY$. Chaque fois que INKEY$ est exécutée, l'ordinateur "scrute" le clavier une fois pour voir si vous appuyez sur une touche. Si oui, INKEY$ renvoie le caractère représenté par la touche – une lettre "A" à "Z" ou un chiffre "0" à "9". Remarquez que INKEY$ traite les chiffres en tant que caractères, pas en tant que nombres. C'est une distinction de taille. Cela veut dire que l'ordinateur *ne range pas* ces chiffres en tant que nombres, pour les utiliser dans des calculs, mais plutôt en tant que simples caractères, comme n'importe quels autres caractères du clavier. (Donc le stockage des chiffres ne se fait pas en *notation flottante* dans l'instruction INKEY$.)

Puisque nous voulons que l'ordinateur accorde une attention constante aux touches enfoncées du clavier, nous devons écrire l'instruction INKEY$ à l'intérieur d'une boucle :

```
110 IF INKEY$ = "" THEN GOTO 110
```

L'instruction GOTO, qui suit le THEN, rend le contrôle du programme au début de la même ligne ! Cette ligne signifie : "Si INKEYS ne renvoie aucun caractère (cela signifiant qu'on n'a appuyé sur aucune des touches du clavier), revenir au début de la ligne, et exécuter à nouveau l'instruction INKEY$." Tant que vous n'aurez pas appuyé sur une touche, l'ordinateur s'obstinera à exécuter sans cesse la ligne 110. Mais dès que vous aurez appuyé sur une touche, l'expression vraie ou fausse :

```
INKEY$ = ""
```

deviendra fausse, et le contrôle du programme passera à la ligne suivante.

Puisque vous avez déjà fait fonctionner ce programme, vous savez que l'instruction INKEY$ diffère complètement de l'instruction INPUT quant à ce qu'elle fait. INPUT écrit tous les caractères que vous introduisez dans le coin en bas à gauche de l'écran. INPUT attend également que vous ayez appuyé sur la touche NEW LINE (ou ENTER), avant de ranger l'information introduite dans une variable. INKEY$ ne fait rien de tout cela; elle lit simplement un unique caractère depuis le clavier, et rend ce caractère au programme.

Une fois que l'instruction INKEY$ a fourni un caractère, la ligne 120 range ce caractère dans la variable chaîne de caractères I$:

```
120 LET I$ = INKEY$
```

Puis, les lignes 130, 150 et 160 testent la valeur de I$ – par trois instructions IF distinctes – pour décider de ce qu'il faudra faire ensuite. Nous avons déjà vu que la ligne 130 rend le contrôle au début du programme, pour remettre l'écran à blanc.

Les lignes 150 et 160 présentent une possibilité de l'instruction IF, dont nous n'avons pas encore parlé – l'utilisation des mots AND et OR à l'intérieur d'expressions vraies ou fausses. Ces deux mots vous permettent de tester plus d'une égalité ou inégalité dans une seule instruction IF.

La ligne 150, qui teste les touches I ou D, comporte le mot OR :

```
150 IF I$ = "I" OR I$ = "D" THEN GOSUB 300
```

Autrement dit, si la variable I$ contient soit le caractère I, soit le caractère D, alors la ligne 150 appelle le sous-programme de la

ligne 300. Une expression qui contient le mot OR est vraie si *l'un ou l'autre* des termes de l'expression est vrai.

La ligne 160 est la ligne la plus complexe du programme. Elle teste si vous avez appuyé sur une des touches de direction, les touches "1" à "8" :

```
160 IF I$ >= "1" AND I$ <= "8" THEN
                GOSUB (100 * (VAL(I$) + 4))
```

Une expression contenant le mot AND n'est vraie que si *les deux termes* de l'expression sont vrais. Donc, cette ligne n'appelle un sous-programme que si la valeur de I$ se trouve entre le caractère "1" et le caractère "8". L'instruction GOSUB de la ligne 160 est différente des autres GOSUB que nous avions vus jusqu'à présent, dans la mesure où elle *calcule* le numéro de la ligne de sous-programme qu'elle appelle. Nous verrons que ceci permet à la ligne 160 d'appeler l'un des huit sous-programmes différents – et c'est vraiment une caractéristique très puissante. La clé de ceci réside dans l'expression se trouvant après le mot GOSUB :

```
(100 * VAL(I$) + 4))
```

Vous comprendrez mieux un certain nombre de choses sur cette expression une fois que vous aurez lu le Chapitre 4. (En fait, la ligne 160 peut alimenter pour un certain temps votre réflexion. Ne vous en faites pas si vous ne la comprenez pas complètement.) En bref, cette expression fournit un multiple de 100 compris entre 500 et 1200; souvenez-vous que ce sont les numéros des lignes des sous-programmes de direction. Les numéros de lignes sont calculés en trois étapes :

1. La fonction VAL convertit la valeur de I$ de caractère (compris entre "1" et "8") en nombre (compris entre 1 et 8).
2. L'expression ajoute 4 à ce nombre, ce qui donne un nombre compris entre 5 et 12.
3. Le résultat de l'étape 2 est ensuite multiplié par 100, ce qui fournit un nombre compris entre 500 et 1200.

Ce qu'il est important de comprendre, c'est que vous pouvez écrire une seule instruction GOSUB qui appellera un sous-

programme parmi plusieurs. L'instruction peut "décider" quel sous-programme appeler, en fonction d'un calcul faisant intervenir une variable.

Pour nous résumer, les lignes 130, 150 et 160 peuvent produire un appel à un sous-programme, mais à condition seulement que vous appuyiez sur une des touches ayant une signification pour ce programme – E, I, D ou 1 à 8. Si vous enfoncez une autre touche, aucune des instructions IF n'aura pour résultat l'appel d'un sous-programme, et le programme continuera simplement à la ligne 170 :

```
170 GOTO 110
```

Ainsi, de toute manière, le programme revient toujours à l'instruction INKEY$ de la ligne 110.

Maintenant regardons rapidement les sous-programmes, en commençant par ceux de direction. Ces petits sous-programmes possèdent tous la même structure. Chacun d'entre eux a deux instructions LET qui affectent des valeurs aux variables MH et MV. Dans le sous-programme qui déplace le pixel, les valeurs de ces deux variables sont utilisées pour amener les coordonnées actuelles dans la bonne direction. Pour voir comment cela se fait, regardez le sous-programme de direction "NORD-OUEST", à la ligne 500. Il met le mouvement horizontal à –1 et le mouvement vertical à 1 :

```
500 LET MH = – U
510 LET MV = U
```

Vous pouvez voir qu'en retranchant 1 de la coordonnée horizontale du pixel, on le déplacera vers la gauche (c'est-à-dire "vers l'Ouest"). De même, en ajoutant 1 à la coordonnée verticale du pixel, on le déplacera vers le haut (c'est-à-dire "vers le Nord").

Chacun des sous-programmes de direction se termine par l'écriture de la nouvelle direction dans le coin en haut à gauche de l'écran (à l'adresse 0,0) :

```
520 PRINT AT Z,Z; "NO"
```

Aux lignes 350 et 360 du sous-programme de déplacement du pixel, les coordonnées actuelles (CX,CY) sont modifiées par les valeurs de MH et MV :

```
350 LET CX = CX + MV
360 LET CY = CY + MV
```

La ligne 350 peut être interprétée comme : "Ajouter la valeur de MH à celle de CX et rangez le résultat dans la variable CX." L'ancienne valeur de CX est perdue, de même que l'ancienne valeur de CY à la ligne 360.

Une fois que les nouvelles coordonnées ont été calculées, le pixel peut être affiché à l'écran :

```
370 PLOT CX,CY
```

Le sous-programme de la ligne 300 possède deux caractéristiques que vous devez étudier. Les lignes 300 à 330 testent les coordonnées actuelles pour voir si le pixel n'a pas atteint l'un des quatre bords de l'écran. Si c'est le cas, la valeur de MH ou de MV doit être modifiée pour ramener le pixel dans la zone d'écran. Par exemple, si la coordonnée horizontale atteint 63 (le côté droit de l'écran), MH, la variable de mouvement horizontal, doit être mise à la valeur –1, pour déplacer le pixel vers la gauche :

```
300 IF CX = 63 THEN LET MH=–U
```

Enfin, à la ligne 340, le sous-programme de déplacement du pixel regarde s'il doit effacer le pixel actuel, avant de placer le nouveau pixel. Si vous avez appuyé sur la touche D, l'instruction UNPLOT efface le pixel de coordonnées CX,CY :

```
340 IF I$ = "D" THEN UNPLOT CX,CY
```

Si vous avez appuyé sur la touche I, l'instruction UNPLOT n'est pas exécutée, et l'ancien pixel reste sur l'écran près du nouveau.

En étudiant soigneusement ce programme, vous pouvez approfondir votre connaissance d'un certain nombre d'instructions BASIC parmi lesquelles LET, IF et GOSUB. De plus, ce programme introduit pour vous plusieurs nouveaux mots du vocabulaire BASIC : REM, INKEY$, TAB, AND, OR et VAL. Si vous trouvez que

cela fait trop de choses à assimiler en une fois, revenez-y plus tard, par exemple lorsque vous aurez lu le dernier chapitre de cet ouvrage.

RÉSUMÉ

Nous avons fait beaucoup de chemin dans ce chapitre. Voici une liste qui vous aidera à vous rappeler des instructions BASIC que vous avez apprises, et à revoir ce qu'elles font :

- Définir des variables et leur affecter des valeurs : LET, INPUT
- Lire des informations depuis le clavier : INPUT, INKEY$
- Afficher des informations à l'écran : PRINT, AT, TAB, CLS
- Afficher des graphismes à l'écran : PLOT, UNPLOT
- Répéter des instructions, créer des boucles : FOR, TO, STEP, NEXT, GOTO
- Contrôler l'ordre de l'exécution : GOTO, GOSUB, RETURN
- Prendre des décisions : IF, AND, OR, THEN
- Documenter votre programme : REM.

Chapitre 4

RECETTE NUMÉRO CINQ : LES NOMBRES SUR VOTRE ORDINATEUR

INTRODUCTION

Dans ce chapitre, vous apprendrez à utiliser votre ordinateur pour faire des calculs numériques. Le chapitre commence par un exposé sur les opérations arithmétiques, puis continue par un rapide survol de quelques-unes des fonctions arithmétiques les plus utilisées. Vous les verrez à l'œuvre sur plusieurs exemples.

Deux programmes utiles vous seront également présentés dans ce chapitre. Le premier transforme votre ordinateur en un "super calculateur", livré avec opérations, fonctions et mémoire. Le deuxième programme, le plus long de ce livre, vous permettra de créer des histogrammes pour n'importe quel type d'informations numériques. Vous apprendrez à utiliser ces deux programmes, et vous aurez l'occasion d'accroître votre connaissance de la programmation en BASIC par l'étude de la structure et de la logique de ces programmes. En particulier, vous apprendrez ce que sont les *tableaux* dans le deuxième programme. Les tableaux sont une structure essentielle de la programmation en BASIC, qui vous permet de ranger de nombreux éléments d'informations sous un seul nom de variable.

LES CALCULS AVEC VOTRE ORDINATEUR

Les opérations arithmétiques que votre ordinateur peut réaliser sont l'addition, la soustraction, la multiplication, la division et l'élévation à une puissance. Les symboles utilisés pour ces opérations sont :

+	addition
–	soustraction
*	multiplication
/	division
**	élévation à une puissance

Un des moyens dont vous disposez pour dire à l'ordinateur de faire un calcul est l'instruction PRINT. L'ordinateur commencera par faire le calcul, puis affichera le résultat à l'écran. Pour voir comment cela se passe, introduisez les différentes lignes suivantes en tant que commandes en mode immédiat :

```
PRINT 15.193 + 6.5
PRINT 1235 - 872
PRINT 18 * 97
PRINT 183/5
PRINT 9 ** 3
```

A chaque fois que vous introduirez l'une de ces commandes, vous verrez le résultat du calcul s'afficher, presque immédiatement en haut de l'écran. La dernière de ces cinq instructions PRINT est un exemple d'élévation à une puissance. L'expression "9 ** 3" signifie "9 à la puissance 3", ou bien "9 au cube". Le résultat, vous pouvez le voir, est 9 × 9 × 9, ou 729. (Le symbole "**" est un caractère majuscule situé sur la touche H.)

Vous pouvez également écrire des opérations arithmétiques à plusieurs termes en une seule instruction. Par exemple, l'expression suivante aura pour résultat la somme de deux produits :

```
PRINT 9 * 8 + 5 * 3
```

Le nombre affiché à l'écran est 87, la somme de 72 et 15. Remarquez que l'ordinateur fait d'abord les deux multiplications,

puis en additionne les résultats. Ceci est un point essentiel. Lorsque l'ordinateur fait l'évaluation d'expressions complexes, il le fait dans un certain ordre. Vous devez toujours être conscient de cet ordre, sinon vous courrez le risque d'obtenir des résultats erronés.

Voici l'ordre des opérations :

1. l'élévation à une puissance, de gauche à droite;
2. la multiplication et la division, de gauche à droite;
3. l'addition et la soustraction, de gauche à droite.

Pour expérimenter cet ordre de priorité des opérations, introduisez la ligne suivante :

```
PRINT 1 + 2 ** 3 * 4 – 5
```

Aviez-vous prévu le bon résultat, 28 ? Voici comment l'expression arithmétique est évaluée; d'abord l'élévation à la puissance, 2 ** 3, ce qui donne 8; puis la multiplication, 8 * 4, dont le résultat est 32; enfin, l'addition de 1 et la soustraction de 5, ce qui donne 28.

Evidemment, si vous aviez introduit cette ligne en pensant que l'ordinateur allait faire les calculs de gauche à droite – indépendamment de la nature des opérations – vous auriez été bien surpris ! En faisant chaque opération de gauche à droite, on aurait obtenu 103. (Essayez : $1 + 2 = 3$; $3^3 = 27$; $27 \times 4 = 108$; $108 - 5 = 103$.) Sans aucun doute, vous devez toujours réfléchir soigneusement à la priorité des opérateurs pour l'ordinateur, avant d'écrire une expression arithmétique.

Cependant, il arrivera souvent que vous vouliez faire des calculs dans un ordre différent de celui-ci. Heureusement, le BASIC vous donne un moyen de passer outre à l'ordre standard de priorité des opérateurs, et d'imposer l'ordre que vous souhaitez dans les calculs. Pour cela, vous devez utiliser des parenthèses, à l'intérieur des expressions arithmétiques.

Lorsqu'une sous-expression est mise entre parenthèses, elle est évaluée en premier. Par exemple, entrez successivement ces deux commandes :

```
PRINT 1 + 2 * 3
PRINT (1 + 2) * 3
```

Dans la première expression, c'est la multiplication qui est effectuée d'abord, puis l'addition et cela donne 7. Dans la deuxième expression, l'ordinateur commence par évaluer l'addition entre parenthèses, puis multiplie le résultat de l'addition par 3; on obtient 9.

Vous pouvez aussi écrire des expressions à plusieurs niveaux de parenthèses. (On appelle souvent *parenthèses imbriquées* des parenthèses à l'intérieur d'autres parenthèses.) L'ordinateur traite d'abord les parenthèses *les plus internes* et ensuite les autres. L'expression suivante, avec ses parenthèses, sera évaluée de gauche à droite :

```
PRINT (((1 + 2) ** 3) * 4) – 5
```

Le résultat de cette expression, comme nous l'avons déjà vu, sera 103. Remarquez quelque chose d'important relatif à l'imbrication des parenthèses : chaque symbole de parenthèse ouvrante "(" doit correspondre à un symbole de parenthèse fermante ")". Si vous écrivez accidentellement une expression ne comportant pas le même nombre de parenthèses ouvrantes et fermantes, l'ordinateur refusera de faire le calcul.

Nous avons déjà vu que les expressions arithmétiques pouvaient comporter des variables; l'expression ne sera valide que si les variables qu'elle contient ont déjà été définies. Ceci s'applique aux instructions LET, PRINT et à la plupart des autres instructions BASIC faisant intervenir des valeurs numériques. Pour vérifier par vous-même que c'est vrai, introduisez les lignes suivantes en tant que commandes en mode immédiat :

```
LET A = 1
LET B = A + 2
LET C = B ** 3
LET D = C * 4
LET E = D – 5
PRINT A, B, C, D, E
```

Chaque instruction LET de cette liste utilise la valeur d'une variable définie dans l'instruction précédente. L'instruction PRINT affiche enfin les valeurs de toutes les variables. Remarquez que la dernière variable E contient la valeur 103, résultat de l'expression

(((1 + 2) ** 3) * 4) − 5. Les autres variables contiennent les valeurs des résultats intermédiaires.

L'instruction PRINT ci-dessus montre une nouvelle propriété du BASIC dont nous n'avions pas encore parlé : l'utilisation de *virgules* pour séparer les éléments d'un affichage. La virgule indique à l'ordinateur d'afficher les éléments en les séparant par une demi-largeur d'écran à blanc. Voici ce que donnera l'instruction PRINT :

```
1                     3
27                    108
103
```

La différence dans l'instruction PRINT entre un point-virgule et une virgule est très importante. Le point-virgule, vous vous le rappelez, indique à l'ordinateur d'afficher les éléments côte à côte.

Vous devez connaître une règle supplémentaire intervenant dans les calculs : la division par zéro n'est pas définie pour l'ordinateur; si vous écrivez une expression provoquant une division par zéro, vous aurez un message d'erreur. Le sachant, vous ne pourrez pas écrire volontairement une expression telle que :

```
LET A = 1/0
```

Mais des problèmes se présentent souvent quand vous écrivez des divisions avec des variables au dénominateur; par exemple :

```
LET B = 1/C
```

Si jamais la variable C comporte la valeur zéro au moment où l'instruction LET est exécutée, l'ordinateur affichera un message d'erreur. Ceci peut être un problème épineux dans un programme pour lequel vous n'êtes pas certain à l'avance de la valeur que prendra C. Une façon d'éviter le problème est d'écrire l'instruction LET en tant que clause d'une instruction IF :

```
IF C <> 0 THEN LET B = 1/C
```

Cette formulation vous assure que l'instruction LET ne sera exécutée que si la variable C comporte une valeur différente de zéro.

Vous avez vu que le clavier de votre ordinateur vous offre un certain nombre de fonctions mathématiques usuelles. Vous pouvez les introduire en mettant le clavier en mode fonction. Examinons l'utilisation de ces fonctions dans les calculs.

LES FONCTIONS ARITHMÉTIQUES

Nous envisagerons les fonctions suivantes :

- ABS (fonction valeur absolue; située au-dessous de la touche G);
- INT (fonction partie entière; située au-dessous de la touche R);
- SGN (fonction signe; située au-dessous de la touche F);
- SQR (fonction racine carrée; située au-dessous de la touche H).

Vous pouvez trouver par vous-même l'utilisation de ces fonctions, même si vous n'avez aucun penchant pour les mathématiques. Votre ordinateur offre également d'autres ensembles de fonctions pour des utilisations spécifiques. Il y a les fonctions trigonométriques (SIN, COS, TAN, ARCSIN, ARCCOS, ARCTAN) et les fonctions exponentielles (EXP, LN). Si ces fonctions vous intéressent, l'Appendice A vous documentera sur elles.

La fonction "valeur absolue", ABS, fournit la valeur *positive* de tout nombre. Autrement dit, si le nombre est négatif, ABS rend un nombre positif. S'il est positif, ABS rend le même nombre sans modification. Introduisez les deux commandes ci-dessous pour voir fonctionner ABS :

```
PRINT ABS – 45
PRINT ABS 45
```

Pour les deux commandes, le résultat est 45. Remarquez que le signe moins de la première expression désigne uniquement le signe de 45, et *non* une opération entre deux nombres différents.

La fonction INT élimine la partie fractionnaire (s'il y en a une) d'un nombre décimal. Par exemple, introduisez ces commandes :

```
PRINT INT 19.34
PRINT INT 19.81
```

Le résultat des deux est 19. Remarquez que la fonction INT ne calcule pas l'arrondi; elle ne fait que *tronquer* à l'entier inférieur. Nous verrons plus loin dans ce chapitre de quelle façon utiliser la fonction INT pour créer une fonction d'arrondi.

La fonction SGN a toujours pour résultat l'une de ces trois valeurs, qui dépendent du signe du nombre que vous écrivez après SGN :

–1	si le nombre est négatif
0	si le nombre est nul
1	si le nombre est positif.

L'expression :

```
PRINT SGN –45, SGN 0, SGN 45
```

fournit la suite des nombres –1, 0 et 1.

Enfin, SQR donne la racine carrée d'un nombre positif. Par exemple :

```
PRINT SQR 81
```

aura pour résultat 9. Si vous introduisez un nombre négatif après SQR, l'ordinateur affichera un message d'erreur; la racine carrée d'un nombre négatif n'est pas définie.

On peut construire des expressions dans lesquelles les fonctions sont les éléments d'opérations arithmétiques; mais une fois de plus, vous devez faire attention à l'ordre des opérations. Les fonctions sont évaluées avant les cinq opérations arithmétiques (à moins que ces opérations ne soient spécifiées entre parenthèses). Par exemple, introduisez l'expression :

```
PRINT 10 * SQR 81 + 19
```

Le résultat en est 109. L'ordinateur calcule d'abord la racine carrée de 81, ce qui donne 9; puis il multiplie 10 par 9, ce qui

donne 90; enfin il additionne 90 et 19, et on obtient 109. Comparez ceci avec l'expression :

```
PRINT 7 * SQR (81 + 19)
```

On obtient 70. L'ordinateur fait d'abord l'addition entre parenthèses, 81 plus 19 donnent 100; puis en extrait la racine carrée; racine carrée de 100 est égale à 10; puis fait la multiplication : 7 que multiplie 10 donne 70.

Enfin, les fonctions peuvent fournir leur résultat à d'autres fonctions, c'est-à-dire que vous pouvez utiliser ce genre d'expression complexe :

```
SQR INT 9.73
```

L'expression dit : "Trouver la racine carrée de la partie entière de 9.73". Si vous introduisez cette expression dans votre ordinateur à l'intérieur d'une instruction PRINT, vous verrez que le résultat sera 3.

Dans le paragraphe suivant, nous compléterons nos connaissances sur l'utilisation de l'instruction INPUT pour faire des calculs, et exécuterons un petit programme qui nous aidera à expérimenter cette possibilité.

FAIRE DES CALCULS AVEC L'INSTRUCTION INPUT

Introduisez NEW pour effacer de la mémoire de votre ordinateur toute ligne de programme; puis introduisez ce programme de trois lignes :

```
10 INPUT V
20 PRINT V
30 GOTO 10
```

A première vue cela ne semble pas être un programme bien utile. La ligne 10 lit une valeur numérique au clavier, et range cette valeur dans la variable V. La ligne 20 affiche le contenu de la

variable V à l'écran. La ligne 30 rend le contrôle du programme à la ligne 10, réalisant une boucle sans fin. Quand vous exécuterez ce programme, l'ordinateur attendra que vous ayez introduit un nombre au clavier, puis affichera ce nombre à l'écran.

Comme nous le verrons, l'ordinateur peut effectivement lire des expressions arithmétiques au clavier pendant l'exécution d'un programme. Ceci est une caractéristique, unique et très utile, du BASIC du ZX81. Au lieu de ne taper que des nombres au clavier, vous pouvez taper n'importe quelle expression arithmétique valide, en utilisant les cinq opérations et les fonctions numériques. L'ordinateur fera immédiatement les calculs puis en rangera le *résultat* dans la variable de l'instruction INPUT, V dans notre cas.

Exécutez maintenant le programme et essayez-le. Commencez par introduire un nombre simple, disons 10; le nombre apparaîtra en haut de l'écran. Vous pouvez voir que le programme fonctionne.

Puis introduisez une expression. Essayez celle-ci :

```
10 * SQR 49 + 16/8
```

Le nombre 72 sera affiché en haut de l'écran. l'ordinateur a évalué l'expression et rangé le résultat, 72, dans la variable V.

Entrez maintenant l'expression :

```
V/9
```

L'ordinateur répondra en affichant le nombre 8 à l'écran. Votre expression V/9 signifiait : "diviser la valeur actuelle de la variable V par 9". Puisque V avait la valeur 72 à la suite du calcul précédent, le résultat de la division est 8.

En résumé, nous avons vu que l'ordinateur acceptera toute expression numérique valide en réponse à une instruction numérique INPUT. Cette expression peut également mettre en œuvre des variables, pour autant qu'elles aient été définies auparavant.

Examinons maintenant le programme qui transforme votre ordinateur en un super calculateur.

LA RÉALISATION D'UN SUPER CALCULATEUR

Le programme SUPER CALCULATEUR apparaît à la Figure 4.1. Introduisez-le soigneusement dans votre ordinateur, puis exécutez-le. Vous verrez le message :

SUPER CALCULATEUR

en vidéo inversée et le caractère indicateur L entre guillemets, ce qui indiquera que l'ordinateur attend qu'on introduise une chaîne de caractères.

Vous pouvez utiliser ce programme pour résoudre n'importe quel calcul. L'expression que vous introduisez peut comporter :

- les cinq opérations arithmétiques;
- n'importe quelle fonction mathématique de votre clavier;
- des expressions complexes comportant plusieurs niveaux de parenthèses, ou des fonctions imbriquées;
- deux variables : M1 et M2, constituant la fonction de mémorisation du programme. (Nous étudierons rapidement cette fonction.)

Autrement dit, vous pouvez introduire n'importe quel calcul accepté par l'ordinateur comme expression numérique valide. Le programme a la forme d'une boucle sans fin; vous pouvez l'utiliser pour autant de calculs que vous voulez.

Essayons-le. Introduisez l'expression suivante au clavier :

```
1 + INT (SQR (2 * 3) * 4)
```

Appuyez sur NEW LINE (ou ENTER), et regardez ce qui se passe. L'écran reste blanc quelques instants. Puis quand les informations y reviennent, votre écran devient tel qu'à la Figure 4.2. Etudiez soigneusement l'écran pour en tirer tout ce qu'il vous dit.

D'abord, l'expression que vous avez introduite au clavier est "reproduite" à votre intention au milieu de l'écran. Ceci est important; il arrive souvent qu'après avoir fait une longue série

d'opérations sur un calculateur de poche, vous vous demandiez nerveusement quand le résultat apparaît, si vous ne vous êtes pas trompé en appuyant sur les touches. Les erreurs d'introduction constituent certainement la cause la plus fréquente d'erreurs avec une calculatrice. Aussi, la première chose que ce programme vous indique, avant même la réponse à votre problème est l'expression exacte que vous avez introduite dans l'ordinateur. Vous pouvez étudier à loisir cette "reproduction" de ce que vous avez saisi, pour être certain que c'est bien le calcul que vous vouliez exécuter. Après cela, l'ordinateur vous donnera la solution du problème. Dans notre cas, la réponse est 10.

De plus, le programme prévoit à votre intention deux variables de mémorisation. Elles s'appellent M1 et M2; après un calcul leurs valeurs courantes sont toujours affichées dans le coin en haut à

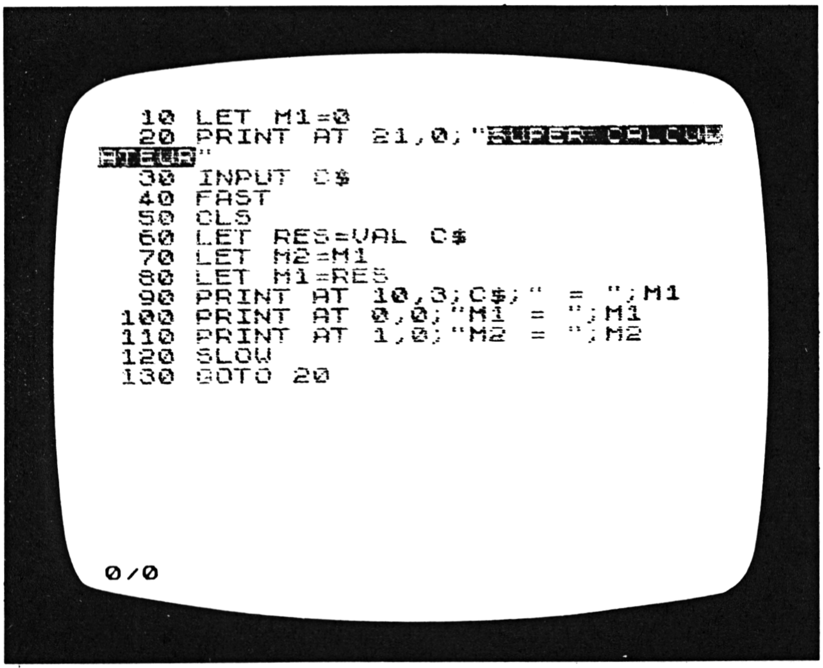


Figure 4.1 : Le programme SUPER CALCULATEUR

gauche de l'écran. M1 est le résultat du dernier calcul fait par l'ordinateur, M2 celui du calcul précédent. Comme vous pouvez le voir à l'écran, les valeurs actuelles de M1 et M2 sont :

M1 = 10
M2 = 0

La valeur de M1 est le résultat du calcul qui est affiché maintenant au milieu de l'écran. Puisque c'est le premier calcul que vous ayez fait pendant cette exécution du programme, M2 est encore égale à zéro. (Au début du programme, M1 et M2 ont toutes deux pour valeur zéro.)

Introduisons maintenant une deuxième expression à évaluer par le programme :

(18 ** 2) / 5

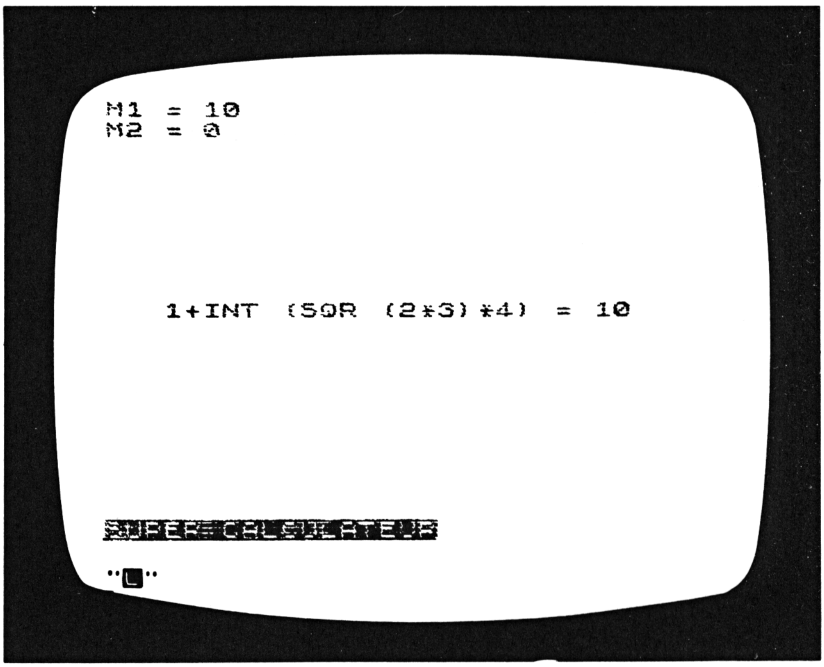


Figure 4.2 : Une réponse du programme SUPER CALCULATEUR

La réponse, que nous voyons à la Figure 4.3, est 64.8. Regardons ce que sont devenues les deux variables mémoire :

M1 = 64,8
M2 = 10

Le dernier résultat apparaît maintenant dans M1. Le résultat du premier calcul, 10, apparaît dans M2. L'objet de ces deux mémoires est le suivant : vous pouvez les utiliser dans des expressions que vous introduirez dans l'ordinateur. Par exemple, introduisez l'expression suivante en tant que troisième calcul :

M1/M2

Cela signifie : "divise la valeur actuelle de M1, par la valeur actuelle de M2". L'ordinateur vous répondra :

M1/M2 = 6,48

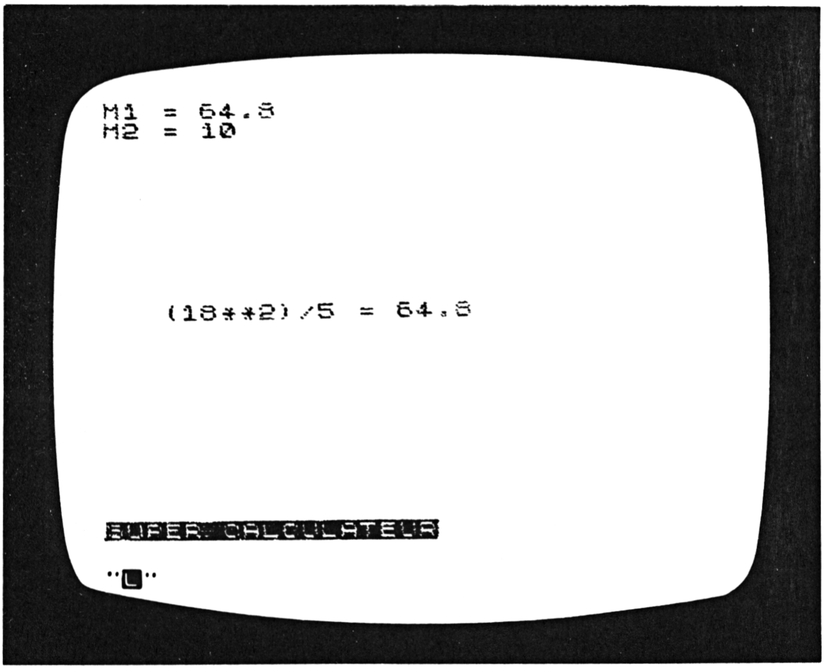


Figure 4.3 : Un second calcul

L'utilisation de ces deux variables sera souvent très utile, quand vous aurez une longue série de calculs à faire. Leurs valeurs actuelles seront modifiées et affichées à chaque étape.

Habituez-vous à ce programme jusqu'à ce que vous ayez maîtrisé complètement son fonctionnement. Il se peut que vous vous aperceviez assez rapidement que si vous introduisez une expression ne pouvant pas être évaluée par l'ordinateur, le déroulement du programme sera interrompu et un message d'erreur affiché. Ceci peut se produire dans un certain nombre de cas; n'importe laquelle des erreurs suivantes provoquera l'interruption du programme :

- introduction d'une expression ne comportant pas le même nombre de parenthèses ouvrantes et fermantes;
- insertion dans l'expression d'une variable inconnue (c'est-à-dire une variable différente de M1 et M2);
- un calcul faisant intervenir une division par zéro;
- l'utilisation d'une fonction n'ayant pas un nombre pour résultat.

Si vous faites une de ces erreurs, vous pourrez naturellement revenir au programme en exécutant simplement la commande RUN. Mais l'inconvénient, si vous faites cela, est que vous perdez les valeurs actuelles des variables mémoire M1 et M2. (La commande RUN, vous vous en souvenez, efface toutes les variables existantes avant de commencer l'exécution du programme.) Une meilleure façon de redémarrer ce programme est d'introduire la commande en mode immédiat :

```
GOTO 100
```

En le faisant, M1 et M2 apparaîtront inchangées à l'écran, et le programme sera prêt pour votre nouveau calcul. Nous verrons exactement comment fonctionne cette commande lors de l'examen de la structure interne du programme. En général, vous devez toujours redémarrer un programme par l'instruction GOTO plutôt que par RUN, de manière à conserver les valeurs actuelles de toutes les variables en mémoire.

Regardons maintenant le programme lui-même. Vous pouvez introduire la commande STOP pour terminer l'exécution du

programme, puis appuyer à nouveau sur la touche NEW LINE (ou ENTER) pour afficher ce dernier à l'écran. Vous pouvez également revenir à la Figure 4.1 qui comporte la liste du programme.

À L'INTÉRIEUR DU PROGRAMME DU SUPER CALCULATEUR

Les trois premières lignes du programme sont claires. La ligne 10 *initialise* la variable mémoire M1 à zéro; la ligne 20 affiche la ligne "Titre du programme" à l'écran; et la ligne 30 lit ce que l'on introduit au clavier. Remarquez que ce que l'on introduit est stocké dans la variable chaîne de caractères C$:

```
30 INPUT C$
```

L'expression *n'est pas* encore évaluée en tant que calcul à faire, ce qui aurait été le cas s'il y avait eu une variable numérique. Tout ce que l'ordinateur sait à ce stade, c'est que C$ comporte une chaîne ordinaire de caractères.

La ligne 40 met l'ordinateur dans le mode rapide :

```
40 FAST
```

Nous avons déjà vu deux fois cette instruction FAST dans les programmes précédents de ce livre, mais nous n'avons jamais dit ce qu'elle faisait exactement. En mode rapide, l'ordinateur cesse d'envoyer des informations sur l'écran TV pendant tout le temps des calculs. Cela a pour résultat de rendre les calculs plus rapides. Lors de l'exécution d'un programme vous pouvez savoir que vous êtes en mode rapide si l'écran devient blanc pendant certaines parties du programme. Dans le programme SUPER CALCULATEUR, rappelez-vous que l'écran deviendra blanc dès que vous aurez introduit l'expression à calculer. L'ordinateur consacre ses ressources uniquement au calcul, et cela permet de terminer le travail plus vite.

A titre d'essai, vous pourriez supprimer la ligne 40, puis exécuter le programme, pour voir la différence. Vous trouveriez que les calculs sont nettement plus longs. (Assurez-vous de bien remettre la ligne 40 du programme avant de continuer.)

La ligne 50 remet l'écran à blanc par l'instruction CLS. C'est vraiment à la ligne 60 du programme que le principal du travail est fait :

```
60 LET RES = VAL C$
```

Comme nous l'avons vu au Chapitre 3, la fonction VAL détermine l'équivalent numérique d'une chaîne de caractères. Pour pouvoir être prise en compte par la fonction VAL, il va de soi que la chaîne de caractères ne doit être composée que de caractères pouvant être convertis en nombres; c'est pourquoi les expressions que vous introduisez dans ce programme doivent toutes être des expressions numériques valides. Si vous faites une faute en introduisant une expression quelconque non valide, vous obtiendrez le message d'erreur :

```
C/60
```

Le 60 indique que le programme s'est terminé à la ligne 60. Comme vous pouvez le voir à l'Appendice B, le code d'erreur C indique que vous avez fourni une chaîne de caractères non valide pour la fonction VAL.

Si tout se passe bien, l'instruction LET de la ligne 60 range le résultat du calcul de votre expression numérique dans la variable numérique RES. Les lignes 70 et 80 modifient les valeurs des variables mémoire M1 et M2. D'abord, on met dans M2 le résultat du calcul précédent, qui était stocké dans M1 :

```
70 LET M2 = M1
```

Puis on met dans M1 le résultat du dernier calcul :

```
80 LET M1 = RES
```

Les trois lignes suivantes préparent l'écran. La ligne 90 affiche l'expression de départ (encore stockée dans la variable chaîne de caractères C$) et le résultat (stocké dans M1) :

```
90 PRINT AT 10,3; C$; " = "; M1
```

Les lignes 100 et 110 affichent le contenu des mémoires dans le coin en haut à gauche de l'écran :

```
100 PRINT AT 0,0; "M1 = "; M1
110 PRINT AT 1,0; "M2 = "; M2
```

C'est pour cela qu'il faut utiliser la commande :

```
GOTO 100
```

pour relancer le programme après une éventuelle erreur ayant provoqué l'arrêt du programme.

Enfin, la ligne 120 remet l'ordinateur en mode lent, puis la ligne 130 rend le contrôle du programme à la ligne 20, ceci constituant une boucle sans fin.

Avant de passer au dernier point de ce chapitre, faites avec le programme SUPER CALCULATEUR une dernière expérience. Introduisez l'expression :

```
5 ** 20
```

au clavier, puis appuyez sur la touche NEW LINE (ou ENTER). Vous avez certainement deviné à l'avance que l'expression "5 à la puissance 20" donnerait un nombre très élevé, mais ce qui s'affichera à l'écran vous surprendra peut-être. Voici :

```
5 ** 20 = 9.5367432E + 13
```

Le résultat est exprimé en "notation scientifique". L'ordinateur passe automatiquement dans cette notation pour les nombres très grands, tels que celui-ci. Cette notation n'est pas aussi difficile à lire qu'elle peut le paraître lorsqu'on n'est pas encore familiarisé avec elle. La lettre E, presque à la fin de l'expression signifie l'exposant, et l'expression E + 13, veut dire 10 élevé à la puissance 13, ou :

10 000 000 000 000.

Ainsi l'expression complète qui a été donnée en notation scientifique vaut :

9.5367432 × 10 000 000 000 000

ou encore :

95 367 432 000 000

Quand vous vous rendez compte que l'ordinateur peut produire des nombres beaucoup plus grands que celui-ci, vous

pouvez commencer à comprendre l'utilité de la notation scientifique. C'est simplement la façon la plus "économique" d'afficher de grands nombres à l'écran.

Si vous êtes intéressé par ce sujet, et que vous vouliez pousser plus avant, vous pouvez utiliser le programme SUPER CALCULATEUR pour répondre aux questions qui vous viennent à l'esprit. En quelques essais, vous pouvez découvrir beaucoup de choses quant à la façon dont votre ordinateur manipule les nombres :

- A quel moment l'ordinateur commence-t-il à afficher les nombres en notation scientifique, au fur et à mesure qu'ils deviennent plus grands ?
- Quel est le plus grand nombre que votre ordinateur peut manipuler ?
- Est-ce que l'ordinateur affiche "exactement" les très grands nombres ? (Essayez d'introduire 5 ** 15. Vous savez que n'importe quelle puissance de cinq doit se terminer par un cinq. Est-ce le cas pour la réponse de l'ordinateur ?)

CRÉATION D'HISTOGRAMMES

Le programme suivant que nous examinerons a pour objet de créer des histogrammes sur votre écran de télévision. Un histogramme fournit un moyen pratique et souvent spectaculaire de comparaison entre des séries de nombres. Chaque nombre est représenté par un trait; plus grand est le nombre, plus long est le trait. En regardant un histogramme vous pouvez, d'un seul coup d'œil, voir l'importance relative de chacune des catégories représentées à l'écran. C'est pourquoi vous pouvez utiliser le programme d'histogrammes pour présenter des informations chaque fois que vous êtes plus intéressé par les valeurs *relatives* des données, que par leurs valeurs individuelles.

Puisqu'un histogramme s'occupe de listes de données, il est temps pour vous d'étudier la structure qu'offre le langage BASIC pour manipuler de telles listes. On appelle cette structure un *tableau.* Nous allons étudier les tableaux et la façon de les définir, avant de passer au programme.

Les tableaux

Vous savez maintenant qu'une variable simple ne peut comporter qu'une valeur à la fois. Si vous affectez une nouvelle valeur à une variable, en utilisant LET ou INPUT, l'ancienne valeur sera perdue à jamais. Les variables simples sont un bon moyen de stockage pour des nombres ou des éléments d'information indépendants les uns des autres, à ceci près qu'ils sont tous nécessaires au programme.

Cependant, il arrivera souvent qu'en utilisant votre ordinateur pour traiter des données, vous ayez des listes ou des tableaux de données, formant des ensembles et que vous vouliez les ranger dans la mémoire de l'ordinateur. On peut avoir besoin de douzaines ou de centaines d'éléments d'information, si bien que ce ne serait pas pratique du tout de devoir définir une variable différente pour chacun d'entre eux.

Le BASIC prévoit des structures de données spéciales, appelées *tableaux,* qui conviennent tout à fait pour stocker de grandes quantités de données. On dit que les tableaux ont des *dimensions.* On utilise les tableaux à une dimension pour stocker des *listes* d'informations. On peut utiliser les tableaux à deux dimensions pour stocker des *tableaux* d'informations, organisés en lignes et colonnes. Sur votre ZX81 vous aurez sans doute rarement besoin de tableaux à plus de deux dimensions, bien que votre BASIC autorise des tableaux multidimensionnés. Le programme d'histogrammes utilise des tableaux à une dimension, aussi nous ne parlerons que du stockage de *listes* de données.

Supposez que vous ayez une liste de 15 nombres à introduire dans la mémoire de votre ordinateur. Vous pouvez concevoir un programme qui exécutera trois tâches :

1. lire les nombres depuis le clavier pendant l'exécution du programme;
2. faire des calculs sur ces nombres;
3. afficher à nouveau ces nombres à l'écran, à côté des résultats du calcul.

Vous pouvez commencer par définir un tableau contenant les 15 nombres, en une seule fois. Disons que vous décidez d'appeler

ce tableau N (comme *Nombres*). Voici comment définir ce tableau, à la première ligne de votre programme :

```
10 DIM N(15)
```

Le mot clé DIM, qui se trouve au-dessus de la touche D, veut dire "dimension". Une instruction DIM précise les quatre caractéristiques essentielles d'un tableau :

1. le nom du tableau;
2. le type d'informations qu'il contiendra (numériques ou de type chaîne de caractères);
3. son nombre de dimensions;
4. le nombre d'éléments d'information qu'il peut contenir, pour chacune des dimensions. (Ce nombre est appelé taille du tableau.)

Voici les caractéristiques du tableau défini plus haut, à la ligne 10 :

1. son nom est N;
2. c'est un tableau numérique (puisque son nom ne se termine pas par le caractère $);
3. il possède une seule dimension (puisqu'il n'y a qu'un seul nombre spécifié entre parenthèses);
4. la taille du tableau est 15.

Quand l'ordinateur arrive à cette ligne du programme, il réserve automatiquement en mémoire de la place pour 15 nombres, référencés par le nom de tableau N.

Une fois le tableau défini, vous avez besoin d'un moyen pour identifier chacun des 15 éléments. Une fois que vous aurez affecté des valeurs aux éléments du tableau, vous voudrez par exemple accéder au 7ème nombre de la liste; c'est pourquoi vous aurez besoin de pouvoir dire exactement à l'ordinateur quel élément vous intéresse.

Le système permettant d'identifier les éléments d'un tableau est très simple. Vous écrivez le nom du tableau, puis entre parenthèses, le numéro de l'élément que vous voulez indiquer. Par exemple, le 7ème élément du tableau N est appelé :

```
N(7)
```

De la même façon les cinq premiers éléments sont :

```
N(1)
N(2)
N(3)
N(4)
N(5)
```

Les nombres entre parenthèses sont appelés les *indices* du tableau. Vous pouvez écrire des instructions du genre :

```
LET N(7) = 162
INPUT N(5)
PRINT N(2)
```

La première instruction range le nombre 162 dans le 7ème poste du tableau N. La deuxième lit un nombre depuis le clavier et le range dans le 5ème poste de N. La troisième instruction affiche à l'écran la valeur actuelle du deuxième élément du tableau N.

Vous pouvez également écrire un nom de variable à l'intérieur des parenthèses qui suivent le nom du tableau; par exemple :

```
N(I)
```

Alors, le choix de l'élément dépend de la valeur de l'indice I. Si I est égal à 5, N(I) indique le cinquième élément de N. Sachant tout cela, vous pourrez utiliser des boucles FOR à l'intérieur desquelles la variable de contrôle sera également utilisée en tant qu'indice d'un tableau. Par exemple :

```
10 DIM N(15)
20 FOR I = 1 TO 15
30 INPUT N(I)
40 NEXT I
```

Dans ce programme, la variable de contrôle, I, est également utilisée en tant qu'indice pour le tableau N. Par ces simples lignes, vous pouvez dire à l'ordinateur de lire quinze nombres depuis le clavier, et de les stocker dans le tableau de nom N. Vous commencez peut-être à mesurer la puissance des outils consistant en tableaux et boucles FOR.

Pour voir la puissance de ces instructions, tapez les lignes 10 à

40, données plus haut. Puis ajoutez-y les 3 lignes qui suivent, qui afficheront les 15 éléments de N à l'écran :

```
50 FOR I = 1 TO 15
60 PRINT N(I)
70 NEXT I
```

Exécutez le programme. Introduisez quinze nombres quelconques dans l'ordinateur; quand ce sera fait, ils seront affichés tous en une fois à l'écran. Il est important de se rappeler que ces quinze nombres sont stockés dans la mémoire de l'ordinateur, et que vous pouvez y accéder et les utiliser pour ce que vous voulez. N'importe quelle opération devant concerner une suite d'éléments peut être intégrée à l'intérieur d'une boucle FOR-NEXT.

Votre ordinateur vous permet également de définir des tableaux de chaînes de caractères, mais c'est un peu plus complexe que pour les tableaux numériques. Une fois de plus, nous examinerons un tableau à une seule dimension. Dans l'instruction DIM, après avoir spécifié la dimension et la taille du tableau, vous devez en plus spécifier à l'intérieur des parenthèses un autre nombre. Ce nombre représente la longueur, le nombre de caractères, de chaque élément du tableau de chaînes de caractères. Les éléments d'un tableau de chaînes de caractères ont tous la même longueur prédéfinie. Regardons un exemple.

Supposons que vous ayez une liste de 15 mots que vous voulez stocker dans la mémoire de l'ordinateur, dans le but de vous en servir. Le mot le plus long contient huit caractères. Vous pouvez définir un tableau de chaînes de caractères pour cette liste de la façon suivante :

```
10 DIM W$(15,8)
```

Le nom de ce tableau est W$; il peut contenir 15 chaînes de caractères, de 8 caractères chacune.

Vous pouvez expérimenter l'utilisation de ce tableau, en tapant le petit programme que voici :

```
10 DIM W$(15,8)
20 FOR I = 1 TO 15
30 INPUT W$(I)
40 PRINT W$(I);
50 NEXT I
```

Remarquez tout d'abord, que vous pouvez spécifier les éléments de ce tableau de chaînes de caractères, de la même façon que vous l'aviez fait pour les tableaux numériques. Le septième mot du tableau W$ est appelé :

```
W$(7)
```

et le 1er mot est appelé :

```
W$(I)
```

Maintenant, exécutez le programme et regardez bien ce qu'il fait. D'abord introduisez un mot faisant exactement huit caractères :

```
SINCLAIR
```

Le mot sera stocké dans W$(1) et, du fait de la ligne 40, il sera également affiché en haut de l'écran. Puis, entrez deux mots devant être rangés respectivement dans W$(2) et W$(3); choisissez des mots de moins de huit caractères :

```
LIVRE
CRAYON
```

Le premier mot apparaîtra à l'écran, juste après le mot SINCLAIR. (Remarquez que la ligne 40 se termine par un point-virgule, qui évite le saut à la ligne.) Le mot CRAYON apparaît cependant trois espaces après le mot LIVRE. Sur l'écran vous verrez ceci :

```
SINCLAIRLIVRE   CRAYON
```

Pourquoi y a-t-il trois espaces entre LIVRE et CRAYON ? Parce que vous avez défini les éléments du tableau W$, comme ayant huit caractères. Puisque LIVRE ne comporte que cinq caractères, l'ordinateur complète W$(2) par des blancs pour en obtenir huit, la bonne longueur.

Essayez maintenant d'introduire un mot ayant plus de 8 caractères :

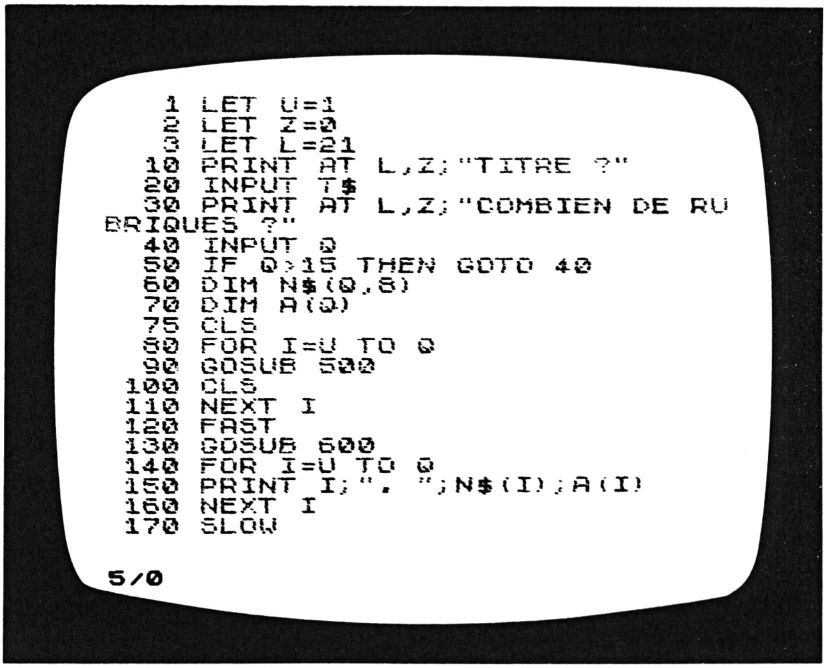
```
ORDINATEUR
```

Vous ne verrez que les huit premiers caractères, ORDINATE, apparaître à l'écran. Le mot tout entier ne tenant pas dans un élément du tableau W$, les deux derniers caractères sont tout simplement éliminés.

Maintenant que vous avez lu cette brève introduction aux tableaux de nombres et aux tableaux de chaînes de caractères, vous êtes prêt à examiner le programme d'histogrammes. Ce programme utilise deux tableaux distincts – définis dans deux instructions DIM – pour stocker les informations nécessaires à la création d'un histogramme.

LE PROGRAMME HISTOGRAMME

La liste du programme d'histogrammes apparaît sur les Figures 4.4 à 4.7. Quand vous introduisez dans votre ordinateur un long programme, comme celui-ci, c'est une très bonne idée de



```
   1 LET U=1
   2 LET Z=0
   3 LET L=21
  10 PRINT AT L,Z;"TITRE ?"
  20 INPUT T$
  30 PRINT AT L,Z;"COMBIEN DE RU
BRIQUES ?"
  40 INPUT Q
  50 IF Q>15 THEN GOTO 40
  60 DIM N$(Q,8)
  70 DIM A(Q)
  75 CLS
  80 FOR I=U TO Q
  90 GOSUB 500
 100 CLS
 110 NEXT I
 120 FAST
 130 GOSUB 600
 140 FOR I=U TO Q
 150 PRINT I;".  ";N$(I);A(I)
 160 NEXT I
 170 SLOW

5/0
```

Figure 4.4 : Le programme d'histogrammes

le stocker sur cassette au fur et à mesure de la saisie. Dès que vous avez saisi environ le quart du programme, sauvegardez-le. Puis continuez jusqu'à la moitié du programme. Une fois que vous aurez effectué la deuxième sauvegarde, vous pourrez réenregistrer à l'endroit où vous aviez sauvegardé la première partie du programme puisque vous n'avez plus du tout besoin de la première partie. Continuez de sauvegarder au fur et à mesure, jusqu'à ce que vous ayez saisi puis sauvegardé le programme tout entier. Cette méthode vous met à l'abri d'une perte éventuelle de votre travail. A tout moment, vous ne pouvez perdre que ce que vous avez saisi depuis la précédente sauvegarde.

Une fois que vous aurez saisi tout le programme, vérifiez soigneusement que vous n'avez pas fait d'erreurs de saisie. Puis lancez l'exécution du programme.

Ce programme crée un histogramme pour une liste quelconque de nombres comportant un maximum de quinze nombres. La

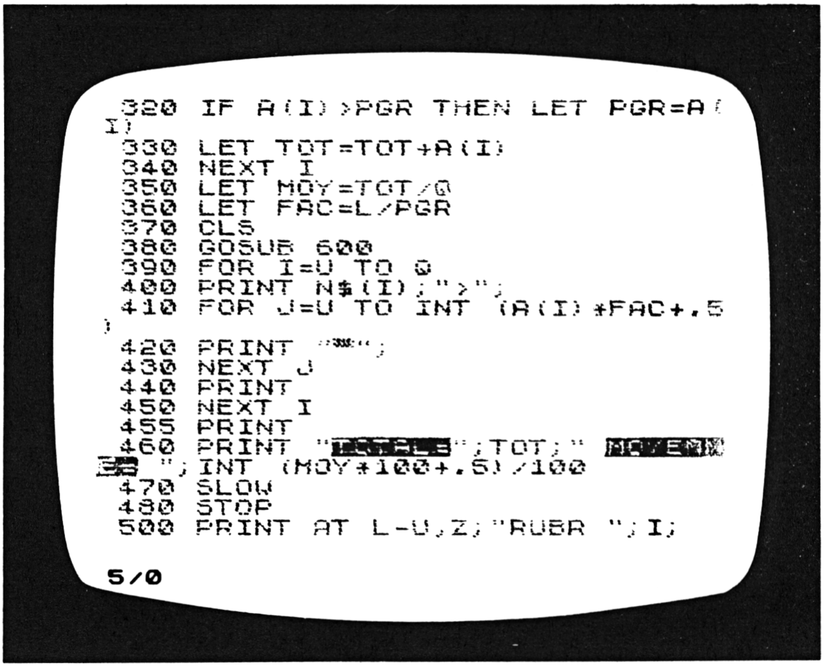
```
 180 PRINT AT L-U,Z;"CORRECTION
? <O> OU <N>"
 190 INPUT A$
 200 IF A$<>"O" THEN GOTO 280
 210 PRINT AT L,Z;"QUEL NO DE RU
BR ?"
 220 INPUT I
 230 IF I>Q THEN GOTO 220
 240 PRINT AT L-U,Z;
 245 GOSUB 700
 250 GOSUB 700
 260 GOSUB 500
 261 PRINT AT I+U,Z;
 263 GOSUB 700
 265 PRINT AT I+U,Z;I;". ";N$(I)
;" ";A(I)
 270 GOTO 180
 280 FAST
 290 LET PGR=Z
 300 LET TOT=Z
 310 FOR I=U TO Q

5/0
```

Figure 4.5 : Le programme d'histogrammes (suite)

Figure 4.8 montre un exemple de résultat final du programme. Cet histogramme particulier compare des dépenses de nourriture, poste par poste, pour le mois de décembre. Si c'étaient les données réelles de votre foyer, cela vous offrirait une représentation graphique facile à lire, de vos dépenses du mois. (Si vous vous cassez la tête en vous demandant à quoi pourrait bien vous servir un histogramme de vos dépenses de nourriture, arrêtez de vous tracasser. Ce programme peut être utilisé pour *n'importe quel* ensemble de données, concernant n'importe quel sujet. Au début du programme, vous fournissez le titre de l'histogramme, le nom de chacune des rubriques, et à côté, sa valeur numérique.)

Ce programme vous guide au travers des trois étapes ayant pour but la réalisation de l'histogramme; d'abord, vous devez introduire toutes les données – comprenant le titre, les noms de rubriques, et les données numériques. Le programme vous



```
 320 IF A(I)>PGR THEN LET PGR=A(
I)
 330 LET TOT=TOT+A(I)
 340 NEXT I
 350 LET MOY=TOT/Q
 360 LET FAC=L/PGR
 370 CLS
 380 GOSUB 600
 390 FOR I=U TO Q
 400 PRINT N$(I);">";
 410 FOR J=U TO INT (A(I)*FAC+.5
)
 420 PRINT "▒";
 430 NEXT J
 440 PRINT
 450 NEXT I
 455 PRINT
 460 PRINT "TOTAL ";TOT;" MOYENNE
 ";INT (MOY*100+.5)/100
 470 SLOW
 480 STOP
 500 PRINT AT L-U,Z;"RUBR ";I;

5/0
```

Figure 4.6 : Le programme d'histogrammes (suite)

fournit des messages indicateurs clairs pour vous dire à tout moment de la phase d'introduction des données quelle information il faut saisir. La seconde phase consiste en la correction de toutes les erreurs de saisie. Cette phase commence par l'affichage, sous forme de tableau, de toutes les informations que vous avez introduites dans l'ordinateur. Vous pouvez les étudier à loisir, et vous pourrez corriger toutes les erreurs que vous trouverez dans les données. Enfin, lors de la troisième phase, vous verrez s'afficher l'histogramme réalisé à partir de vos données. Examinons chacune de ces trois phases; nous introduirons les données de dépenses de nourriture pendant l'exécution, à titre d'exemple, du programme.

Au début, comme vous l'avez déjà vu à l'écran, le programme demande que vous lui donniez le titre de l'histogramme :

TITRE ?

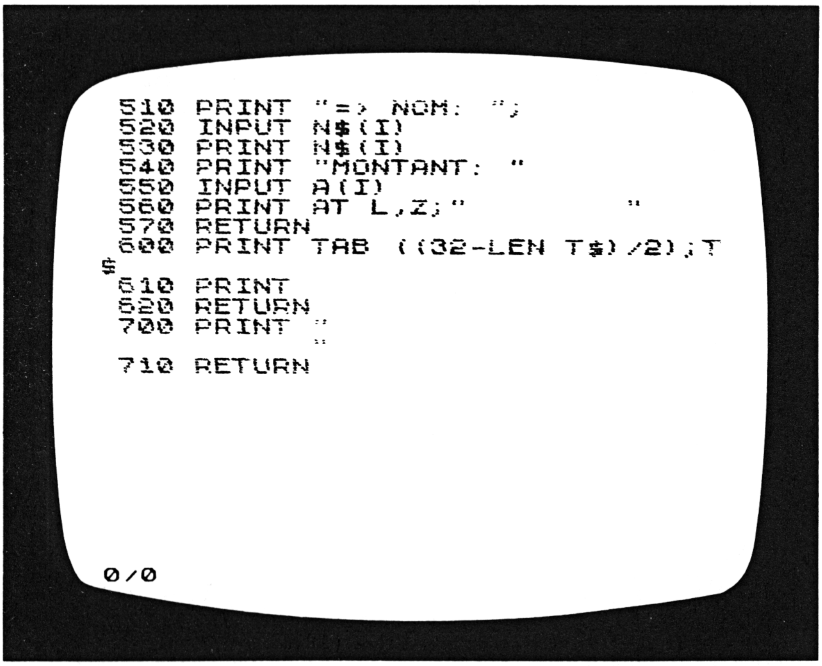


Figure 4.7 : Le programme d'histogrammes (fin)

Vous pouvez entrer n'importe quel titre ayant au maximum 32 caractères. Introduisez le titre du haut de la Figure 4.8. Après ce message, vous verrez une deuxième question s'afficher à l'écran :

COMBIEN DE RUBRIQUES ?

Vous devez introduire le nombre de rubriques de votre histogramme (jusqu'à 15 rubriques). Dans le cas des dépenses de nourriture, il y a 15 rubriques, donc vous devez taper 15 puis appuyer sur NEW LINE (ou ENTER).

Ensuite le programme vous demandera d'introduire le nom et le montant de chacune des 15 rubriques de votre histogramme. Le message indicateur pour la première rubrique sera :

RUBR 1 => NOM:

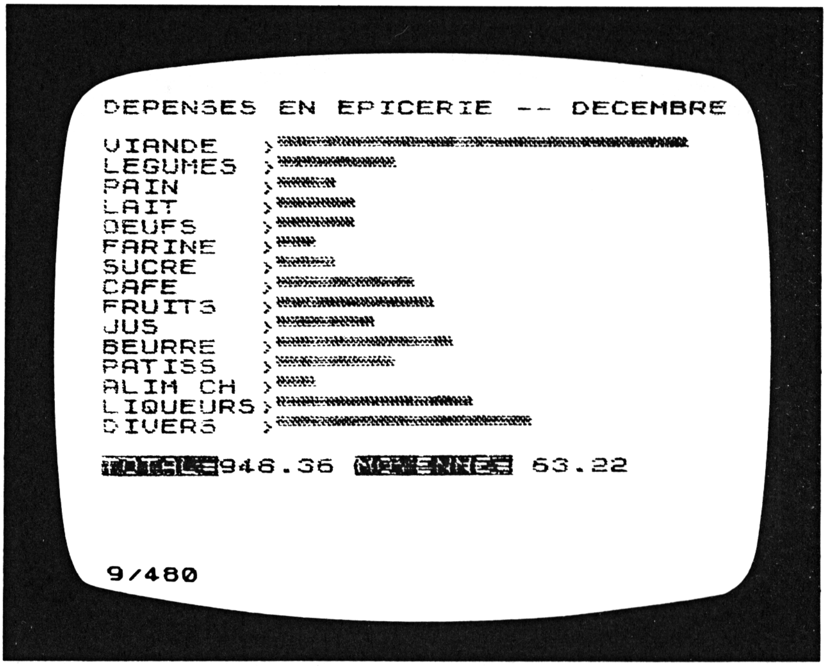


Figure 4.8 : Exemple d'exécution du programme d'histogrammes : Dépenses en épicerie

Il faut introduire le mot VIANDE, qui est la première rubrique nourriture. Dès que vous l'aurez fait, le message indicateur deviendra :

```
RUBR 1 => NOM: VIANDE
MONTANT:
```

Maintenant vous devez introduire la somme que vous avez dépensée en viande ce mois-là. Introduisez le montant 192.71.

Le programme continuera à vous demander le nom et le montant de chacune des autres rubriques, jusqu'à ce que vous ayez tout introduit. La Figure 4.9 présente les rubriques à introduire à la suite de la rubrique VIANDE. (Remarquez que chaque nom ne peut dépasser 8 caractères, donc il faut éventuellement trouver des abréviations claires.)

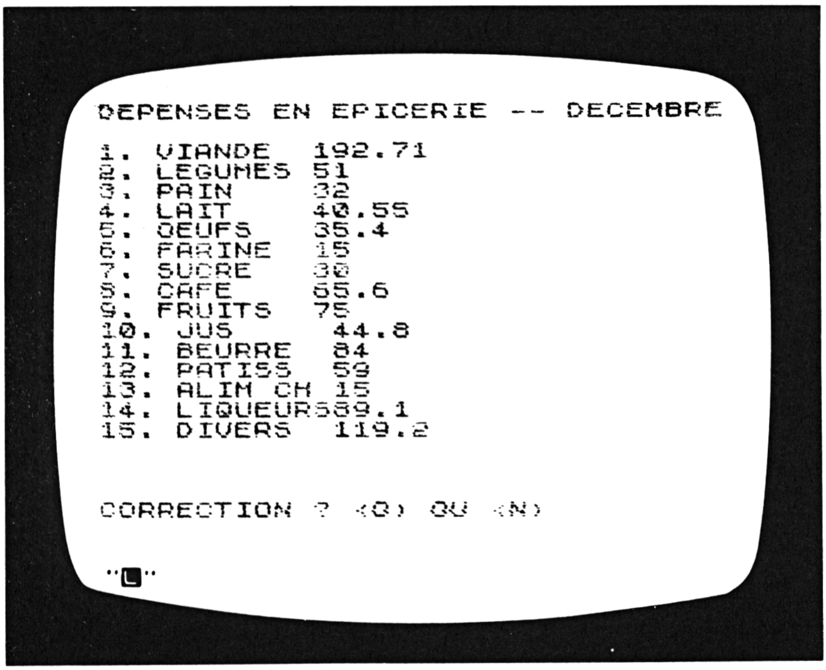


Figure 4.9 : Les données Epicerie

Une fois que vous aurez introduit la dernière rubrique, l'écran deviendra blanc pendant quelques instants. Puis tout ce qui se trouve à la Figure 4.9 apparaîtra sur l'écran. C'est le "rappel" de toutes les informations que vous avez fournies au programme. En bas de l'écran, juste après le tableau de données, apparaît la question :

CORRECTION ? <O> OU <N>

C'est le moment de vérifier les données pour trouver d'éventuelles erreurs de saisie. Si vous trouvez une erreur, vous n'aurez que O (comme "Oui") à taper pour indiquer que vous souhaitez faire une correction.

Comme réponse, l'ordinateur vous demandera :

QUEL NO DE RUBR ?

(en clair : "QUEL NUMERO DE RUBRIQUE ?"). A ce stade, vous devez introduire le numéro de la rubrique que vous souhaitez modifier. Pour voir comment cela se passe, modifions le montant affiché pour la rubrique numéro neuf, la rubrique "FRUITS". Introduisez le nombre 9. Le programme vous demandera alors d'introduire le nom et le montant pour cette rubrique. (Vous devez introduire les deux informations, même si vous ne voulez en modifier qu'une seule.) Introduisez FRUITS pour le nom, puis la nouvelle valeur, 75 comme montant. La correction se fera sur le tableau qui est affiché, ce qui vous permettra de voir immédiatement le résultat de la correction. Vous pouvez effectuer autant de corrections que vous le souhaitez; l'étape de correction ne se terminera que lorsque vous aurez répondu N (comme "Non") à la question.

Dès que vous aurez quitté la phase de correction du programme, l'écran deviendra blanc pour quelques instants, puis l'histogramme apparaîtra à l'écran. Regardez à nouveau l'histogramme des dépenses de nourriture de la Figure 4.8. Remarquez qu'en plus du graphisme, le programme vous fournit le montant total dépensé dans le mois, et le montant moyen dépensé par rubrique.

La Figure 4.10 présente un deuxième exemple d'histogramme.

Cette fois le titre en est : "FOURN. BUREAU 1983" (c'est-à-dire : Fournitures de bureau, pour l'année 1983) et les traits représentent les dépenses mensuelles en fournitures de bureau, de janvier à décembre. Le montant total des dépenses de l'année est, comme on le voit, de 4444.6 Francs, ce qui représente une dépense moyenne mensuelle de 370.38 Francs.

Dans la suite de ce chapitre, nous examinerons la structure de ce programme. Avant de continuer la lecture, il se peut que vous souhaitiez vous servir encore de ce programme, en introduisant des données réelles vous concernant (relatives à votre maison, votre travail ou vos études) ce qui créera un histogramme ayant vraiment un sens pour vous.

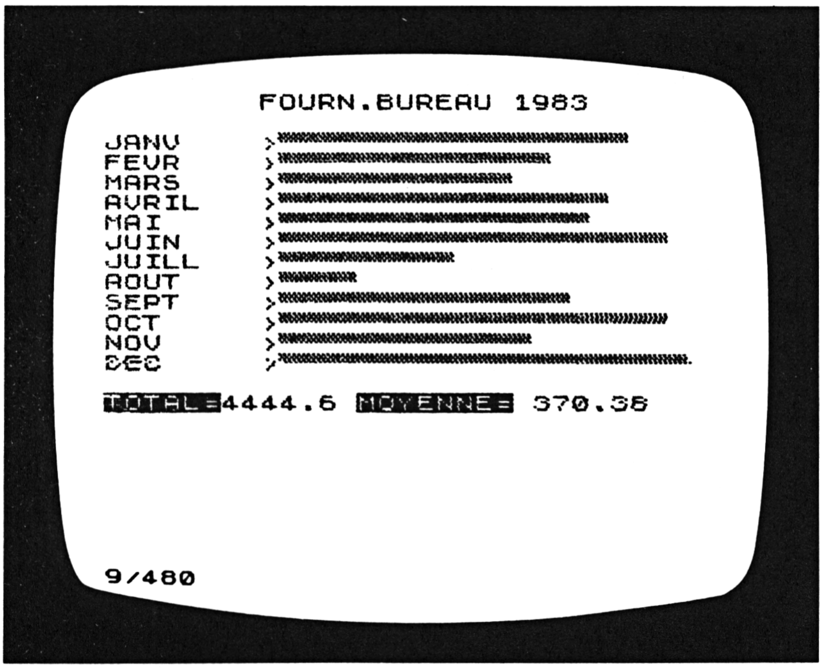


Figure 4.10 : Deuxième exemple de programme d'histogrammes : Fournitures de bureau

À L'INTÉRIEUR DU PROGRAMME D'HISTOGRAMMES

Là aussi, en lisant cette explication, vous pouvez soit vous reporter aux Figures 4.4 à 4.7, où la liste du programme est présente, soit lister le programme morceau par morceau sur votre écran.

Ce programme est encore plus proche que ceux que nous avons déjà vus, de la limite de remplissage de la mémoire, soit 2 K octets pour le TS 1000. Avec le ZX81, il nécessite une extension mémoire. Pour limiter la taille mémoire utilisée, trois variables ont été définies au début du programme, et seront utilisées en tant que "constantes" dans le programme; comme nous l'avons vu au Chapitre 3, remplacer les constantes, fréquemment utilisées par des variables initialisées en début de programme, est un bon moyen pour économiser de la place en mémoire :

```
1 LET U = 1
2 LET Z = 0
3 LET L = 21
```

La variable L est utilisée comme adresse d'affichage des messages à l'écran.

Un autre moyen, pour économiser de la place en mémoire, est de ne pas mettre de lignes REM dans ce programme. C'est dommage pour un programme aussi long que celui-ci; mais c'est un choix qu'il vous faudra parfois faire. Pour compenser ce manque de commentaires dans la liste du programme, en voici la description section par section :

• *Section d'initialisation* (lignes 1 à 70)

Cette section définit les trois variables "constantes"; elle lit le titre de l'histogramme et le nombre de rubriques depuis le clavier; elle réserve les deux tableaux N$ et A, qui serviront à stocker les données de l'histogramme.

- *Section d'introduction des données* (lignes 75 à 110)

Cette section commande l'introduction de toutes les données de l'histogramme. Elle appelle le sous-programme de la ligne 500 une fois pour chacune des rubriques. Le sous-programme de la ligne 500 affiche en retour les messages à l'écran, et lit les données depuis le clavier.

- *Section d'affichage des données* (lignes 120 à 170)

Cette section "montre" toutes les données ayant été introduites, sous forme d'un tableau complet, à l'écran. Elle appelle le sous-programme de la ligne 600 pour afficher le titre en haut de l'écran.

- *Section de corrections* (lignes 180 à 270)

Cette section effectue le contrôle du dialogue de modification. Chaque fois que vous modifiez une rubrique, cette section range la nouvelle donnée dans les postes adéquats des tableaux N$ et A, et l'affiche à l'écran. La section de corrections appelle deux sous-programmes : celui de la ligne 500, qui se charge d'afficher les messages demandant d'introduire les données, puis lit ces dernières au clavier; et le sous-programme de la ligne 700, qui efface une seule ligne de l'écran, pour que les nouvelles informations puissent y être affichées.

- *Section de calculs intermédiaires* (lignes 280 à 360)

Cette section calcule quatre valeurs à partir des données numériques rangées dans le tableau A. PGR (comme Plus GRande) est la plus grande valeur du tableau. TOT (comme TOTal) représente la somme des valeurs du tableau. MOY (comme MOYenne) est la moyenne des valeurs du tableau. Enfin, la valeur FAC (comme FACteur de conversion) représente le facteur de conversion qui sera appliqué dans le but d'obtenir des traits sur l'histogramme, proportionnels aux valeurs qu'ils représentent.

- *Section d'affichage de l'histogramme* (lignes 370 à 480)

Cette section commence par un appel au sous-programme de la ligne 600, qui affiche le titre de l'histogramme. Puis elle trace

chaque trait, en ayant calculé sa longueur grâce au facteur de conversion FAC. Enfin, cette section affiche total et moyenne des données.

• *Les sous-programmes*

– ligne 500 : le sous-programme d'introduction de données;
– ligne 600 : le sous-programme de titre;
– ligne 700 : le sous-programme d'effacement.

En utilisant la description qui vient d'être faite pour vous guider, vous devez être capable de lire le programme dans son entier, et de comprendre ce que font la plupart des lignes. Quelques lignes requièrent cependant des explications supplémentaires. Ce qui suit éclairera quelques-unes des lignes les plus difficiles du programme :

Les lignes 60 et 70 comportent des instructions DIM. La ligne 60 définit le tableau de chaînes de caractères, N$, ayant pour objet le stockage des noms de rubriques, et la ligne 70 définit le tableau numérique, A, qui recevra les montants de chaque rubrique de l'histogramme :

```
60 DIM N$(Q,8)
70 DIM A(Q)
```

La *taille* de ces deux tableaux est spécifiée par la variable Q. Pendant l'exécution du programme, vous introduisez une valeur pour cette variable, en réponse à la question, "COMBIEN DE RUBRIQUES ?" :

```
30 PRINT AT L, Z; "COMBIEN DE RUBRIQUES ?"
40 INPUT Q
```

Si, par exemple, vous introduisez 12 comme nombre de rubriques de l'histogramme, alors la dimension de chacun des tableaux sera fixée à 12, comme si les lignes 60 et 70 étaient :

```
60 DIM N$(12,8)
70 DIM A(12)
```

Au travers de l'utilisation de la variable Q, vous voyez que l'espace mémoire réservé pour les deux tableaux aura exactement la taille nécessaire, ni plus ni moins.

C'est dans la boucle FOR les lignes 80 à 110, que des valeurs seront affectées aux éléments des deux tableaux :

```
 80 FOR I = U TO Q
 90 GOSUB 500
100 CLS
110 NEXT I
```

L'instruction FOR fait varier la variable de contrôle, I, depuis 1 jusqu'à la valeur de Q. Le sous-programme de la ligne 500 utilise cette variable de contrôle pour spécifier le numéro correct de rubrique dans le message qu'il affiche à l'écran :

```
500 PRINT AT L – U,Z; "RUBR";I;
```

et également pour ranger chaque élément d'information dans le poste adéquat du bon tableau :

```
520 INPUT N$(I)
.
.
.
550 INPUT A(I)
```

Ces lignes d'introduction de données, et les lignes de messages qui les précèdent, sont isolées dans des sous-programmes distincts, car on en a besoin à deux moments différents dans le programme : d'abord, dans la section initiale d'introduction des données; puis dans la section de correction.

Une autre boucle FOR, aux lignes 140 à 160, affiche l'ensemble des informations des deux tableaux à l'écran :

```
140 FOR I = U TO Q
150 PRINT I; "."; N$(I); ""; A(I)
160 NEXT I
```

La ligne 150 utilise la variable de contrôle I trois fois; d'abord pour afficher le numéro de la rubrique, puis pour accéder au bon

élément de N$, dans le but d'obtenir le nom de la rubrique; enfin pour afficher le montant de la rubrique, à partir du tableau A.

Les lignes 310 à 340 déterminent le plus grand élément du tableau A (PGR), et la somme des montants (TOT). Les deux variables utilisées pour ces calculs sont d'abord initialisées à zéro :

```
290 LET PGR = Z
300 LET TOT = Z
```

Puis l'autre boucle FOR parcourt le tableau tout entier de 1 à Q :

```
310 FOR I = U TO Q
320 IF A(I) > PGR THEN LET PGR = A(I)
330 LET TOT = TOT + A(I)
340 NEXT I
```

La ligne 320 compare chaque valeur du tableau avec la valeur actuelle de PGR. Chaque fois qu'elle trouve une valeur supérieure à celle de PGR, cette valeur devient le nouveau PGR. La ligne 330 constitue la somme des différentes valeurs.

La valeur moyenne est obtenue en divisant le total par le nombre de valeurs, Q :

```
350 LET MOY = TOT/Q
```

Le facteur de conversion proportionnel est calculé grâce au montant le plus élevé du tableau, PGR :

```
360 LET FAC = L/PGR
```

La variable L, vous vous en souvenez, possède la valeur "constante" 21. Ainsi le plus long trait de l'histogramme aura 20 caractères de long. Comme nous le verrons plus loin, en multipliant chaque valeur du tableau A par le facteur de conversion FAC, on obtiendra un nombre compris entre zéro et 21; ce nombre, à son tour, déterminera la longueur du trait de l'histogramme qui lui correspond.

Le graphe de l'histogramme est affiché sur l'écran aux lignes 390 à 450. Ces lignes méritent d'être étudiées soigneusement. Elles vous montrent un exemple d'une boucle FOR *à l'intérieur* d'une autre boucle FOR. On appelle souvent ceci des boucles *imbriquées;* deux règles essentielles doivent être respectées lorsque vous projetez l'utilisation de boucles imbriquées :

1. La boucle intérieure doit utiliser une variable de contrôle distincte de la boucle extérieure.
2. Les instructions FOR et NEXT de la boucle intérieure doivent se trouver toutes deux à l'intérieur de la boucle extérieure.

Examinez les lignes qui créent le graphisme de l'histogramme, tout en gardant ces règles à l'esprit.

La boucle extérieure, qui commence à la ligne 390, détermine le numéro de rubrique, en faisant progresser la variable de contrôle, I, de la valeur un à la valeur Q :

```
390 FOR I = U TO Q
```

Le nom de la rubrique est d'abord affiché :

```
400 PRINT N$(I); " > ";
```

Puis la boucle intérieure dessine un trait de longueur proportionnelle au montant A(I) :

```
410 FOR J = U TO INT (A(I) * FAC + .5)
```

L'expression A(I) * FAC aura pour résultat, nous l'avons vu, un nombre compris entre zéro et 21. En ajoutant 0.5 et en prenant la partie entière du résultat, nous pouvons arrondir ce nombre à la valeur supérieure :

```
INT (A(I) * FAC + .5)
```

De cette façon, la boucle intérieure fait progresser sa variable de contrôle, J, de un à la longueur appropriée du trait graphique représentant la valeur A(I). Chaque fois que la valeur de J est incrémentée, la ligne 42 augmente d'un caractère la longueur du trait. (Le caractère graphique, qui compose le trait – comme une pile de cubes – se trouve sur la touche S de votre clavier.)

En résumé, vous pouvez voir qu'à chaque répétition de la boucle extérieure, la boucle intérieure travaille à créer un des traits de l'histogramme. C'est probablement la ligne de ce programme la plus importante à étudier. Si vous avez du mal à voir comment cela peut fonctionner, l'analogie qui suit vous aidera à saisir ce concept de boucles imbriquées.

Imaginez le mouvement des trois aiguilles d'une horloge. Si vous définissez le déroulement du temps comme si c'était une "boucle", vous pouvez trouver trois niveaux de boucles sur l'horloge. La boucle la plus interne est l'aiguille des secondes qui fait soixante tours à chaque tour de l'aiguille des minutes. L'aiguille des minutes fait douze tours pendant que celle des heures n'en fait qu'un. Ainsi, on trouve trois niveaux de répétition; c'est la boucle la plus interne qui déploie la plus grande activité, et c'est la boucle la plus externe qui contrôle toute l'activité des deux autres. Dans notre programme, le fonctionnement des boucles imbriquées est similaire : la boucle extérieure de la ligne 390 contrôle le fonctionnement de la boucle interne de la ligne 410.

RÉSUMÉ

Votre ordinateur met à votre disposition cinq opérations arithmétiques (+, −, *, /, **) et un ensemble utile de fonctions mathématiques. Vous pouvez combiner ces opérations et ces fonctions en expressions complexes devant être calculées par l'ordinateur. En écrivant de telles expressions, vous devez toujours avoir à l'esprit la priorité entre opérations prévue par l'ordinateur. Pour passer outre à cet ordre de priorités, ou bien simplifier une expression pour qu'elle devienne plus claire et facile à lire, vous devez ajouter des parenthèses dans cette expression.

Le mode rapide peut être utile pour consacrer toute la rapidité de l'ordinateur à des calculs, comme nous l'avons vu dans le programme SUPER CALCULATEUR.

Les tableaux sont des structures importantes du langage BASIC,

qui permettent de stocker plusieurs valeurs sous un même nom de variable. Un tableau est un outil idéal à l'intérieur d'une boucle FOR : la variable de contrôle d'une boucle FOR peut être utilisée en tant qu'indice de tableau.

Le programme de tracé d'histogrammes présenté dans ce chapitre montre un bon exemple de la puissance des boucles FOR imbriquées, ainsi que plusieurs exemples illustrant l'efficacité de l'usage des tableaux à l'intérieur de boucles FOR.

Chapitre 5

DES MOTS, DES MOTS, DES MOTS : Les chaînes et fonctions caractères sur votre ordinateur

INTRODUCTION

Dans ce dernier chapitre nous étudierons attentivement les chaînes de caractères, ainsi que les fonctions et opérateurs que vous pouvez utiliser pour les traiter. Comme vous le savez, une chaîne de caractères est un élément d'information constitué d'un ou plusieurs caractères. Vous avez déjà acquis une certaine expérience des chaînes de caractères et des variables de ce type au cours des programmes présentés dans les chapitres précédents; il vous reste à découvrir plusieurs fonctions importantes.

Nous commencerons ce chapitre en regardant la codification des caractères utilisée par le ZX81. Vous verrez de quelle façon utiliser les fonctions CODE et CHR$ pour passer des caractères aux nombres les représentant, et vice versa. Vous examinerez et ferez fonctionner deux petits programmes dans ce chapitre; vous apprendrez également ce qu'est la concaténation, et vous vous familiariserez avec cette fonction utile, qu'est la fonction LEN.

Puis avant d'aborder le dernier programme de ce livre, vous découvrirez ce qui arrive quand vous "découpez" une chaîne de caractères, et pourquoi vous êtes susceptible de vouloir le faire. Le dernier et amusant programme, est un "programme d'aide à la décision"; il fait de l'ordinateur votre conseiller particulier. Même si vos questions sont très délicates, il ne vous laissera pas tomber !

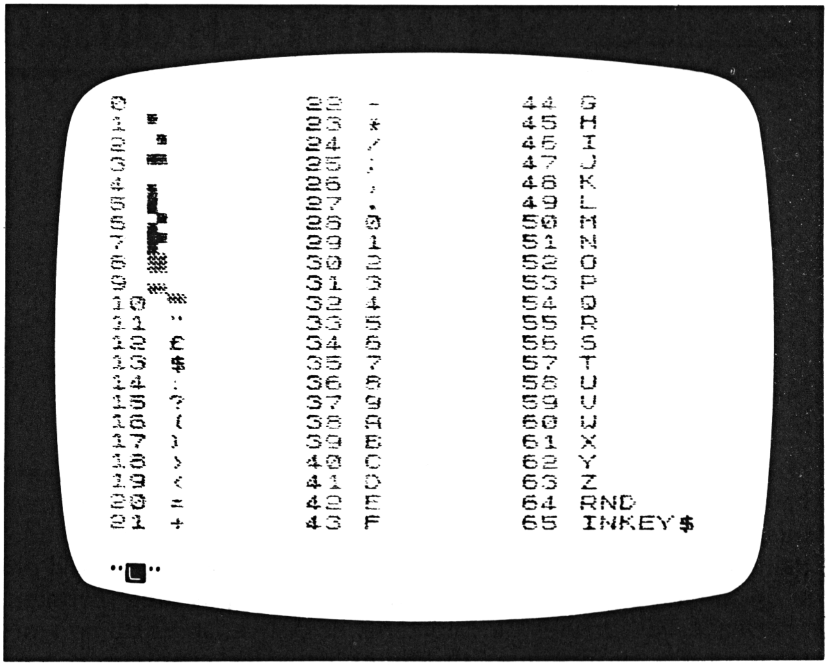


Figure 5.1 : Codification des caractères

LA CODIFICATION DES CARACTÈRES DU ZX81

L'ordinateur utilise sa propre codification pour stocker – et reconnaître – les caractères de votre clavier. Pour un certain nombre de tâches de programmation, il peut vous être utile de connaître cette codification.

Dans cette codification, à chaque caractère est affecté un nombre compris entre 0 et 255. De cette façon, l'ordinateur considère comme un *caractère* chaque élément d'information du clavier – y compris les mots clés, les fonctions et les autres symboles apparaissant sur le clavier. Les Figures 5.1 à 5.3 vous présentent cette codification. Vous n'avez nullement besoin de la savoir par cœur; mais prenez un moment pour vous familiariser

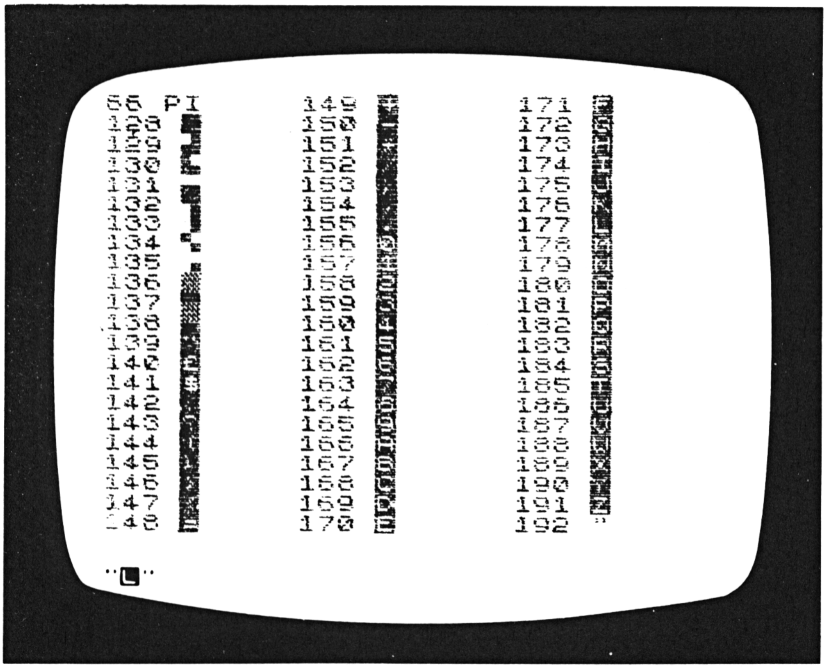


Figure 5.2 : Codification des caractères (suite)

avec elle et remarquer les nombres affectés aux différentes catégories de caractères. Par exemple, vous pouvez voir que les caractères graphiques sont répartis en deux sous-ensembles dans la codification (des valeurs 1 à 10, et 128 à 138); les chiffres et les lettres se trouvent en début de codification (28 à 63); les caractères en vidéo inversée apparaissent à peu près au milieu de la codification (139 à 191). Vous pouvez également retrouver les mots clés, fonctions, opérateurs, et autres symboles dans cette codification.

Vous pouvez remarquer qu'une grande partie de la codification manque (de 67 à 127). La plupart de ces codes numériques manquants sont tout simplement inutilisés; quelques-uns d'entre eux sont réservés aux "touches de contrôle", telles que les touches de directions, et la touche RUBOUT (ou DELETE), qui n'affichent aucun symbole à l'écran.

La fonction CHR$ fournit le caractère représenté par un nombre



Figure 5.3 : Codification des caractères (fin)

donné. Ce nombre doit, naturellement, être compris entre 0 et 255; la fonction CHR$ fournit le caractère correspondant, ou bien un mot clé, une fonction ou un autre symbole.

Les six lignes de programme qui suivent lisent un nombre au clavier et affichent à votre intention à l'écran le caractère qui correspond dans la codification au nombre introduit :

```
10 PRINT AT 21,0; “QUEL NOMBRE DE LA CODIFICATION?“
20 INPUT C
30 IF C > 255 THEN GOTO 20
40 CLS
50 PRINT AT 8,8; “CHR$(“;C;“) = “;CHR$ C
60 GOTO 10
```

La ligne 30 vérifie que le nombre introduit appartient à la codification du ZX81. La ligne 50 utilise la fonction CHR$ pour afficher le caractère.

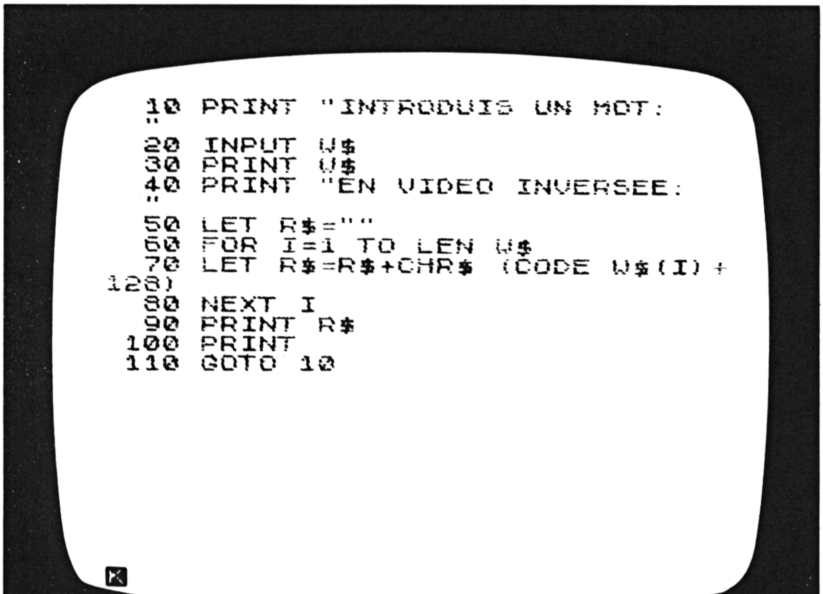


Figure 5.4 : Le programme de vidéo inversée

A l'inverse, la fonction CODE fournit le nombre correspondant à un caractère donné, dans la codification. Rappelez-vous, nous avions utilisé les fonctions CHR$ et CODE au Chapitre 2; le tout premier programme que vous aviez introduit dans votre ordinateur utilisait ces deux fonctions pour mettre en vidéo inversée une chaîne de caractères.

Le programme de la Figure 5.4 met en vidéo inversée une chaîne de caractères. Ce programme n'est qu'un exercice; il ne sert vraiment à rien. Mais vous pourrez apprendre quelques techniques de travail intéressantes sur les chaînes de caractères, en prenant le temps de l'étudier soigneusement.

La Figure 5.5 montre quelques exemples de ce que fait ce programme. Comme vous pouvez le voir, le programme vous dit sans cesse d'introduire une chaîne de caractères dans l'ordinateur. Dès que vous l'avez fait, le programme affiche cette chaîne de caractères en vidéo inversée. Le procédé d'inversion,

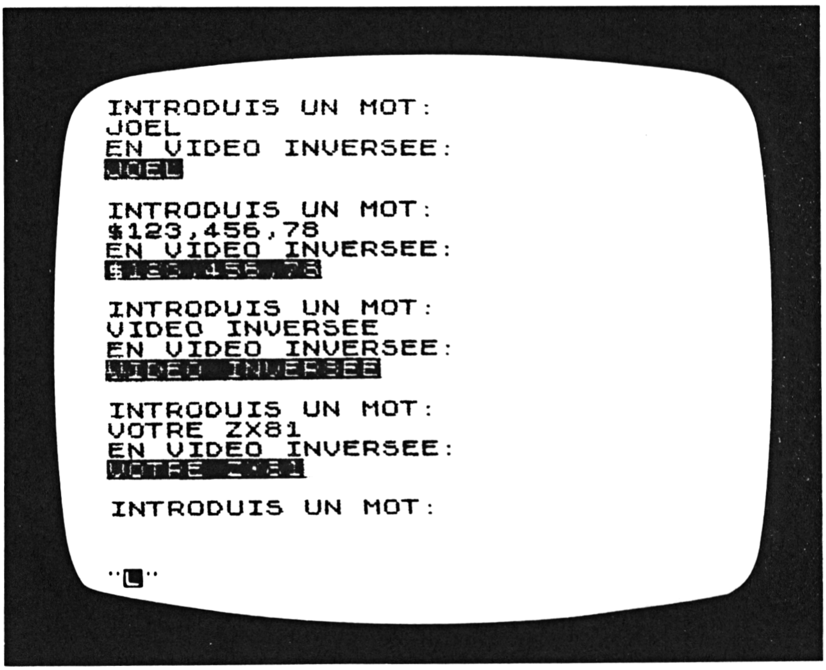


Figure 5.5 : Exemple de sortie du programme vidéo inversée

qui se trouve aux lignes 50 à 80, constitue le seul intérêt de ce programme.

La variable chaîne de caractères W$ reçoit le mot que vous avez introduit. La variable R$ contiendra ce mot en vidéo inversée. A la ligne 50, on initialise R$ comme chaîne de caractères *vide,* c'est-à-dire comme chaîne de caractères ne comportant aucun caractère.

```
50 LET R$ = ""
```

Puis la boucle FOR des lignes 60 à 80 détermine l'équivalent en vidéo inversée de chacun des caractères de W$, et ajoute ce caractère inversé à la fin de la chaîne de caractères R$. L'instruction FOR de la ligne 60 utilise la fonction LEN pour déterminer la longueur, en nombre de caractères, de la chaîne qui a été introduite, W$. Comme vous le voyez, la variable de contrôle I est incrémentée de 1 à LEN W$:

```
60 FOR I = 1 TO LEN W$
```

C'est de cette façon que la boucle traite chaque caractère de W$, un seul à la fois. Le BASIC vous permet d'accéder à *un caractère* d'une chaîne de caractères, en donnant un nombre qui représente la position du caractère, entre parenthèses, après le nom de la variable chaîne de caractères. Ainsi, W$(I) représente le caractère de W$ situé en Ière position.

Par exemple, si vous introduisez une chaîne de trois caractères dans l'ordinateur, la boucle FOR fera varier I de 1 à 3. Les trois caractères de la chaîne seront W$(1), W$(2) et W$(3).

Regardons maintenant l'expression de la ligne 70, qui convertit un caractère en son inverse vidéo :

```
CHR$(CODE W$(I) + 128)
```

A l'intérieur des parenthèses, la fonction CODE fournit le numéro dans la codification du caractère W$(I). Regardons à nouveau la codification des Figures 5.1 à 5.3. Si ce que vous avez introduit consiste en signes de ponctuation, chiffres ou lettres, le numéro dans la codification de chacun des caractères sera compris entre 11 et 63. Si vous ajoutez 128 à n'importe lequel de ces nombres, vous verrez que le numéro résultant correspondra à

l'inverse vidéo du caractère de départ. Voyons cela sur un exemple. Le caractère D a dans la codification le numéro 41. En additionnant 41 et 128 on obtient 169; et c'est bien cela, le caractère D en vidéo inversée correspond à 169 dans la codification.

Vous pouvez voir maintenant exactement ce que réalise l'expression de conversion de la ligne 70. Elle commence par ajouter 128 au numéro dans la codification du caractère en question :

```
(CODE W$(I) + 128)
```

puis elle utilise la fonction CHR$ pour trouver le caractère correspondant au nouveau nombre dans la codification :

```
CHR$(CODE W$(I) + 128)
```

Une fois que ce caractère a été déterminé, il est ajouté à la suite de la chaîne des caractères en vidéo inversée de la variable R$. On appelle *concaténation* cette opération de combinaison de deux chaînes de caractères. Ce processus est symbolisé par l'opérateur + (plus), de la même façon qu'à la ligne 70 :

```
70 LET R$ = R$ + CHR$(CODE W$(I) + 128)
```

Cette instruction dit : "Ajouter le nouveau caractère à la fin de la chaîne de caractères qui se trouve actuellement stockée dans la variable R$; puis ranger la nouvelle chaîne de caractères obtenue, qui a un caractère de plus qu'avant, à nouveau dans R$".

La transformation en vidéo inversée est terminée dès que I atteint la valeur de LEN W$; on sort de la boucle FOR, et la ligne 90 affiche R$ à l'écran :

```
90 PRINT R$
```

En résumé, les lignes 50 à 80 de ce programme ont pu vous apprendre plusieurs notions importantes relatives aux chaînes de caractères et à leur manipulation :

- Vous pouvez initialiser une chaîne de caractères en tant que *chaîne de caractères vide,* ceci signifiant que cette chaîne ne comporte aucun caractère (ligne 50).
- La fonction LEN fournit la longueur, en nombre de caractères, d'une chaîne de caractères (ligne 60).

- Vous pouvez accéder à un caractère précis d'une chaîne en spécifiant la position du caractère entre parenthèses (ligne 70).
- Les numéros des caractères en vidéo inversée, dans la codification, sont tous supérieurs de 128 à ceux des caractères normaux (ligne 70).
- Vous pouvez concaténer deux chaînes de caractères (c'est-à-dire les combiner), en utilisant le symbole "+" (ligne 70).

Dans le paragraphe suivant, vous verrez que l'ordinateur peut accéder à plusieurs caractères consécutifs à l'intérieur d'une chaîne de caractères, par un procédé que l'on appelle *extraction* ou *sélection d'une sous-chaîne.*

EXTRACTION OU SÉLECTION D'UNE CHAÎNE DE CARACTÈRES

Il est souvent commode de pouvoir distinguer une *sous-chaîne* d'une chaîne de caractères plus grande, pour plusieurs raisons. Vous pouvez vouloir examiner une chaîne de caractères, pour voir si elle contient une certaine suite de caractères; vous pouvez également vouloir modifier les valeurs de certaines positions à l'intérieur d'une chaîne de caractères. Dans le programme d'aide à la décision, donné à la fin de ce chapitre, vous n'aurez besoin que *d'accéder* de façon relative à certaines parties d'une longue chaîne de caractères, dans le but de les afficher. La technique d'extraction, ou de sélection d'une sous-chaîne, vous permet de faire tout cela simplement et efficacement.

Nous nous familiariserons avec cette technique d'extraction par une série de commandes en mode immédiat. Assurez-vous que chacune de ces commandes est introduite correctement dans votre ordinateur, afin de bien découvrir le fonctionnement de l'extraction.

D'abord, définissons la chaîne de caractères W$, de la façon suivante :

```
LET W$ = "FONDAMENTAL"
```

Puis introduisez la commande :

```
PRINT W$
```

pour être certain que le mot FONDAMENTAL est bien actuellement stocké en mémoire sous le nom de variable W$.

La syntaxe de l'extraction utilise le mot TO, à l'intérieur des parenthèses, à la suite du nom de la variable chaîne de caractères. De façon générale, l'expression :

```
S$(M TO N)
```

identifie la partie de la chaîne de caractères S$ commençant à la position M et se terminant à la position N de la chaîne. M et N peuvent être des nombres littéraux, des variables numériques, ou des expressions numériques. On appelle souvent ce genre de partie extraite d'une chaîne de caractères, une sous-chaîne de caractères.

D'abord, essayons une sélection en introduisant cette commande :

```
PRINT W$(5 TO 8)
```

Avant d'appuyer sur la touche NEW LINE (ou ENTER), essayez de prévoir quel va être le résultat de la commande, en énumérant les positions des caractères du mot FONDAMENTAL. La commande doit afficher le mot AMEN en haut de l'écran.

Vous pouvez omettre dans la clause TO le premier ou le deuxième nombre de position. Si c'est le premier, l'extraction se fera à partir du premier caractère de la chaîne :

```
PRINT W$(TO 3)
```

Cette commande fournit le mot FON. Si vous omettez le deuxième nombre, l'extraction se fera jusqu'à la fin de la chaîne de caractères. Par exemple :

```
PRINT W$(6 TO)
```

En introduisant cette commande, vous devez voir le mot MENTAL à l'écran.

Vous pouvez également *substituer* une sous-chaîne de

caractères. Pour le faire, vous devez spécifier la sous-chaîne que vous voulez modifier dans une instruction LET, telle que :

```
LET W$(TO 5) = "MALAD"
```

Introduisez cette commande, puis faites :

```
PRINT W$
```

pour voir ce qui se passe à l'écran. Vous devez voir le mot MALADMENTAL. Les cinq premiers caractères de la chaîne de caractères ont été remplacés pour les nouveaux caractères.

Le programme d'aide à la décision utilise cette technique d'extraction pour sélectionner la réponse – à l'intérieur d'une longue chaîne de réponses – à la question que vous lui aurez posée. Le secret décevant de ce programme est le suivant : les réponses sont choisies totalement au hasard. Avant d'étudier le programme, nous devons jeter un rapide coup d'œil à la méthode d'obtention de nombres aléatoires avec votre ordinateur.

LES NOMBRES ALÉATOIRES

La fonction RND, qui se trouve au-dessous de la touche T, sélectionne un *nombre aléatoire* compris entre 0 et 1; elle fournit un nombre différent à chaque fois que vous l'utilisez. Pour voir fonctionner RND, exécutez le programme de deux lignes que voici :

```
10 PRINT RND
20 GOTO 10
```

Une colonne de nombres aléatoires apparaîtra à l'écran; le programme continuera jusqu'à ce qu'il n'y ait plus de place dans la colonne. Les nombres aléatoires sont, en principe, tirés de façon totalement imprévisible. En réalité votre ordinateur utilise une formule mathématique pour *calculer* les nombres fournis par RND. Cela veut dire que ces nombres ne sont pas vraiment aléatoires; cependant la formule utilisée est conçue de telle façon qu'ils *semblent* toujours aléatoires. Pour les programmes que

vous écrirez sur votre ordinateur, cette *apparence* de hasard sera certainement largement suffisante.

Sur les petits ordinateurs personnels, tels que le ZX81, l'utilisation la plus fréquente de la fonction aléatoire est probablement la simulation d'événements aléatoires dans les programmes de jeux. Par exemple, vous pouvez utiliser RND pour simuler un lancer de dés, ou la main qui vous est distribuée en jouant aux cartes.

Pour réaliser de telles simulations nous avons besoin d'une méthode de conversion de *la gamme* des nombres aléatoires générés par RND. Des nombres compris entre 0 et 1 ne servent pas à grand-chose; mais si nous pouvons les convertir en nombres aléatoires entiers, compris entre 1 et 6, par exemple, nous pourrons alors nous servir de ces nombres dans certains jeux.

La formule permettant de réaliser une telle conversion est assez simple. Si vous avez un nombre aléatoire compris entre 0 et 1, et si vous le multipliez par 6, vous obtenez un nombre aléatoire compris entre 0 et 6 :

```
RND * 6
```

Si vous arrondissez ce nombre à sa partie entière (en utilisant la fonction INT) vous aurez alors un nombre entier compris entre 0 et 5 :

```
INT(RND * 6)
```

Enfin, en ajoutant 1 à ce résultat, vous créerez un nombre entier aléatoire compris entre 1 et 6 :

```
INT(RND * 6) + 1
```

Pour voir fonctionner cette formule, introduisez le petit programme de la Figure 5.6. Ce programme crée un jeu très élémentaire. L'ordinateur commence par afficher le message qui suit à l'écran :

```
JE PENSE A UN NOMBRE
COMPRIS ENTRE 1 ET 6, DEVINE LE
```

Vous introduisez le fruit de votre intuition qui se présente sous

la forme d'un entier compris entre 1 et 6. Puis l'ordinateur vous indique si vous avez bien deviné ou non. Par exemple si vous avez pensé à 4 vous aurez l'une des deux réponses suivantes :

C.EST BIEN CELA. C.EST 4

ou :

DESOLE, C.EST 2

Bien que ce jeu soit vraiment très simple, il vous montre comment RND peut être utilisée pour faire créer par l'ordinateur des résultats imprévisibles. Remarquez que la ligne 5 du programme calcule le nombre aléatoire, en utilisant la formule développée plus haut :

```
5 LET R = INT (RND * 6) + 1
```

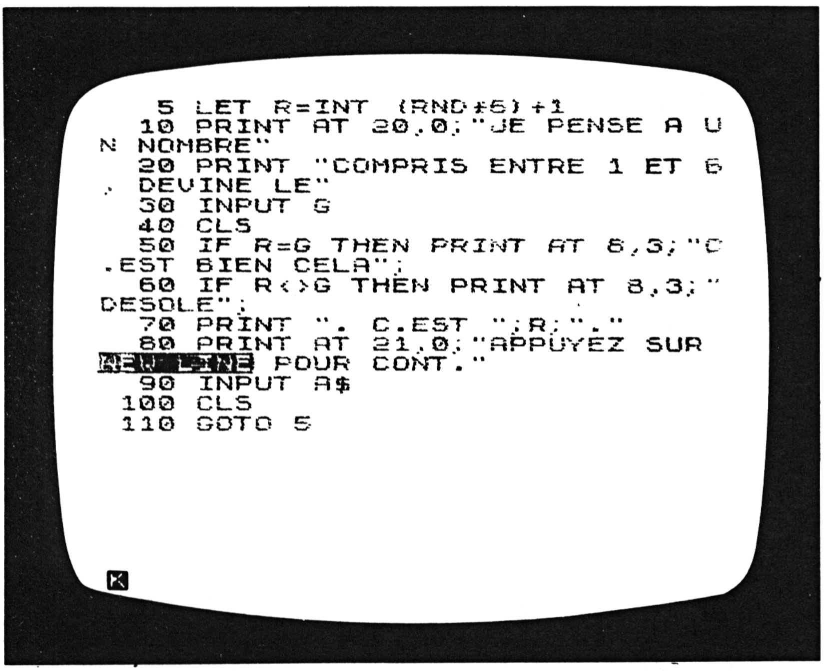


Figure 5.6 : Le programme DEVINETTE

Vous allez voir une formule semblable dans le programme d'aide à la décision :

LE PROGRAMME D'AIDE À LA DÉCISION

Vous pouvez utiliser ce programme en tant que jeu de société, pour amuser vos amis par son étonnante capacité à prévoir l'avenir, résoudre des problèmes personnels ou prendre des décisions professionnelles rapides. Pour utiliser ce programme, vous tapez au clavier n'importe quelle question ayant pour réponse oui ou non; l'ordinateur prend quelques instants pour étudier la question, puis affichera sa réponse "après avoir pesé le pour et le contre" à l'écran.

Naturellement, le secret horrible que vous partagez désormais



Figure 5.7 : Exemple d'affichage obtenu avec le programme AIDE A LA DÉCISION

avec l'ordinateur, est que les réponses sont choisies au hasard, et n'ont aucun rapport avec la nature de la question. Mais sachons garder ce secret.

La Figure 5.7 présente un exemple d'affichage obtenu avec ce programme. Une fois que vous avez introduit une question, l'ordinateur répète la question en haut de l'écran, puis affiche sa réponse à l'intérieur d'un cadre. Puis l'ordinateur attend que vous lui posiez la question suivante; il peut continuer à répondre à des questions toute la journée, si cela vous plaît.

La liste du programme se trouve à la Figure 5.8. Nous aborderons les trois techniques importantes illustrées par ce programme : l'utilisation de RND pour effectuer un choix aléatoire parmi plusieurs réponses; l'utilisation de l'extraction d'une chaîne de caractères pour accéder à la bonne réponse; et enfin, l'utilisation de la fonction LEN pour centrer une chaîne de caractères au milieu de l'écran.

```
  10 LET A$="    OUI        NON
 PEUT-ETRE ABSOLUMENTSI TU VEUXR
EDEMANDE "
  20 PRINT AT 13,1;"PROGRAMME D'
AIDE A LA DECISION"
  30 FOR Y=21 TO 31
  40 PLOT 17,Y
  50 PLOT 45,Y
  60 NEXT Y
  70 FOR X=17 TO 45
  80 PLOT X,21
  90 PLOT X,31
 100 NEXT X
 110 INPUT Q$
 120 IF Q$="" THEN GOTO 150
 130 PRINT AT 2,0;"
"
 140 PRINT AT 2,INT ((32-LEN Q$)
/2);Q$
 150 LET R=10*INT (RND*6)+1
 160 PRINT AT 8,11;A$(R TO R+9)
 170 GOTO 110

K
```

Figure 5.8 : Le programme AIDE A LA DÉCISION

Commencez par étudier soigneusement la ligne 10. Elle définit la "chaîne de réponses" A$. La chaîne a exactement 60 caractères de long et se décompose en six sous-chaînes de 10 caractères, représentant les six réponses différentes que l'ordinateur peut donner à vos questions :

```
OUI
NON
PEUT-ETRE
ABSOLUMENT
SI TU VEUX
REDEMANDE
```

Puisque ces réponses ont des longueurs différentes, elles sont séparées par des espaces à l'intérieur de la chaîne de caractères; ces espaces les complètent à 10 caractères. Ainsi, la première réponse commence en A$(1) et se termine en A$(10); la deuxième commence en A$(11) et se termine en A$(20); la troisième commence en A$(21) et se termine en A$(30) et ainsi de suite. Nous verrons que c'est cette régularité qui permet de sélectionner les réponses à l'intérieur de la chaîne de caractères.

La ligne 20 affiche le titre du programme. Les lignes 30 à 100 dessinent le "cadre de la décision" à l'écran, en utilisant deux boucles FOR et quatre instructions PLOT.

La ligne 110 lit votre question au clavier, et l'affecte à la variable chaîne Q$:

```
110 INPUT Q$
```

Les lignes 150 et 160 choisissent la réponse. La ligne 150 utilise la fonction RND dans une formule étudiée pour l'extraction de la sous-chaîne :

```
150 LET R = 10 * INT (RND * 6) + 1
```

Nous avons vu que cette expression INT(RND * 6) produit un nombre entier compris entre 0 et 5. En le multipliant par 10 on obtient un nombre aléatoire multiple de 10, compris entre 0 et 50. En y ajoutant 1 on obtient l'un des six nombres :

```
1  11  21  31  41  51
```

Ces nombres, comme nous l'avons vu, représentent les adresses respectives de début des six réponses différentes contenues dans la chaîne de caractères A$. Ainsi, la ligne 150 choisit au hasard l'une de ces six adresses de début, et affecte cette adresse à la variable R.

Une fois que cette adresse de début a été déterminée, l'accès pour extraire la réponse est simple. Il est fait à la ligne 160 :

```
160 PRINT AT 8,11; A$ (R TO R + 9)
```

Cette ligne extrait une sous-chaîne de 10 caractères de la chaîne A$, en commençant à la position R et terminant à la position R + 9; puis elle affiche cette sous-chaîne à l'écran, à l'intérieur du cadre. Ainsi, en utilisant la variable R, la ligne 160 affiche une réponse choisie au hasard, sur l'écran. La ligne 170 débute un nouveau cycle en rendant le contrôle du programme à l'instruction INPUT :

```
170 GOTO 110
```

Une ligne supplémentaire de ce programme mérite une explication. La ligne 140, qui affiche votre question en haut de l'écran, comporte une intéressante formule de *centrage* horizontal de la chaîne de caractères Q$:

```
140 PRINT AT 2, INT ((32 – LEN Q$)/2); Q$
```

(Si vous êtes attentif, vous avez déjà pu remarquer une formule similaire dans le programme d'histogramme du Chapitre 4.) La clause AT de cette formule place la chaîne de caractères en ligne 2; la position de la colonne de début est calculée à partir d'une formule basée sur la longueur de la chaîne de caractères :

```
INT ((32 – LEN Q$)/2); Q$
```

Cette formule suppose que la chaîne de caractères de la question, Q$, n'a pas plus de 32 caractères de longueur. La fonction LEN fournit le nombre de caractères de Q$. Puisque l'écran a 32 colonnes en largeur, l'expression (32 – LEN Q$)/2 fournit le numéro de colonne où l'on doit placer la chaîne, pour qu'elle soit centrée horizontalement. Par exemple, supposons que

vous ayez introduit une question de 20 caractères; pour la centrer, il vous faudrait ajouter 6 espaces à droite et à gauche de la question. Vous pouvez voir que la formule (32 – LEN Q$)/2 vous donne la bonne adresse de départ.

RÉSUMÉ

Votre ordinateur possède une codification pour chacun des caractères du clavier : un nombre compris entre 0 et 255. En général vous pouvez programmer en ignorant cette codification; l'ordinateur l'utilise uniquement pour lui, et vous n'avez même pas besoin de savoir qu'elle existe. Mais au cas où cette codification vous serait utile, les fonctions CODE et CHR$ peuvent vous aider pour les conversions entre caractères et codification numérique.

Concaténation signifie combinaisons de chaînes de caractères pour en créer une nouvelle. L'extraction est le procédé qui permet de sélectionner ou de trouver une sous-chaîne de caractères. Nous avons vu des exemples de ces deux notions dans les programmes de ce chapitre.

Maintenant, vous devez être familiarisé avec votre ordinateur, ses opérations et ses possibilités de calcul. Vous êtes prêt à utiliser n'importe lequel des programmes du commerce disponibles pour votre ordinateur, ou à écrire le vôtre. Vous avez une expérience pratique du BASIC, un langage puissant et polyvalent. En bref, vous n'êtes plus seulement un *possesseur* d'ordinateur, vous êtes maintenant un *utilisateur* d'ordinateur.

Appendice A

Le vocabulaire du BASIC

Cet appendice définit les mots qui apparaissent sur le clavier de votre ZX81, et, dans de nombreux cas, donne des exemples d'instructions BASIC faisant intervenir ces mots. L'emplacement sur le clavier de chacun de ces mots est indiqué, à côté de la catégorie du mot (mot clé, fonction, ou majuscule).

ABS (fonction; touche <G>). Fournit la *valeur absolue* d'un nombre.

AND (majuscule; touche <2>). Crée des expressions composées vraies ou fausses pour l'instruction conditionnelle IF. Par exemple :

```
IF I > 0 AND I < 100 THEN GOSUB 300
```

Dans cet exemple, l'instruction GOSUB qui suit THEN ne sera exécutée que si les deux expressions I > 0 *et* I < 100 sont vraies toutes deux.

ARCCOS (fonction; touche <S>). Fournit l'arc cosinus d'un nombre compris entre –1 et 1. Le résultat est exprimé en radians (1 radian = 57,3 degrés). ARCCOS est affiché "ACS" à l'écran.

ARCSIN (fonction; touche <A>). Fournit l'arc sinus d'un nombre compris entre –1 et 1. Le résultat est exprimé en radians. ARCCOS est affiché "ASN" à l'écran.

ARCTAN (fonction; touche <D>). Fournit l'arc tangente, en radians. ARCTAN est affiché "ATN" à l'écran.

AT (fonction; touche <C>). Utilisé avec l'instruction PRINT pour spécifier une position d'affichage à l'écran d'un caractère, d'une chaîne de caractères, ou d'un chiffre. AT doit être suivi d'une adresse sous la forme *ligne, colonne,* où *ligne* est un entier compris entre 0 et 21, et *colonne* un entier compris entre 0 et 31. Par exemple, l'instruction :

```
PRINT AT 8,10; "BONJOUR"
```

affiche le mot BONJOUR à la ligne 8, en commençant en colonne 10. (L'adresse 0,0 représente le coin en haut à gauche de l'écran.)

BREAK (touche <SPACE>). Interrompt l'exécution en cours du programme. BREAK ne peut être utilisé quand l'ordinateur attend l'entrée d'une donnée au clavier.

CHR$ (fonction; touche <U>). Fournit le caractère correspondant à un nombre entier donné, compris entre 0 et 225. Les mots clés, les fonctions, les caractères et les symboles sont tous représentés dans la codification de l'ordinateur.

CLEAR (mot clé; touche <X>). Efface toutes les variables et leurs valeurs de la mémoire; *n'efface pas* les lignes du programme.

CLS (mot clé; touche <V>). Efface toutes les informations présentes sur l'écran; l'instruction PRINT suivante affichera ses informations à partir du coin en haut à gauche de l'écran (la position 0,0).

CODE (fonction; touche <I>). Fournit pour un caractère donné le code interne de l'ordinateur pour ce caractère. L'intervalle des numéros de code va de 0 à 255.

CONT (mot clé; touche <C>). Continue l'exécution d'un programme ayant été interrompue pour une raison quelconque.

COPY (mot clé; touche <Z>). Déclenche l'impression, sur l'imprimante, de tout ce qui se trouve à l'écran.

COS (fonction; touche <W>). Fournit le cosinus d'un nombre représentant un angle. La fonction COS requiert un angle exprimé en radians. (1 degré = 0,0175 radian).

DELETE (majuscule; touche <0>). Supprime le caractère, mot clé, ou fonction se trouvant immédiatement à gauche du curseur. DELETE peut être utilisé chaque fois qu'une ligne est en cours d'introduction dans l'ordinateur, ou quand une ligne complète est "éditée". Présente sur le TS 1000.

DIM (mot clé; touche <D>). Définit le nom, le type, la taille et la longueur d'un tableau. Par exemple :

```
DIM N(10)
```

définit N, tableau de nombres, à une dimension de taille 10. Dans le cas de tableaux de chaînes de caractères, tous les éléments doivent avoir la même longueur qui est spécifiée dans l'instruction DIM. Par exemple :

```
DIM A$(10,5)
```

définit le tableau à une dimension, A$, de chaînes de caractères. A$ peut contenir jusqu'à 10 chaînes, chacune de 5 caractères.

EDIT (majuscule; touche <1>). Affiche une copie de la *ligne courante* en bas de l'écran, pour son "édition". La ligne courante est indiquée, dans la liste du programme par le caractère ">" en vidéo inversée.

ENTER Introduit une ligne de programme ou une donnée dans la mémoire de l'ordinateur. Se trouve sur le TS 1000.

EXP (fonction; touche <X>). Fournit la fonction exponentielle d'un nombre. (Calcule la puissance de e, où e vaut 2,7182818.)

FAST (majuscule; touche <F>). Met l'ordinateur dans le mode "calcul rapide".

FOR (mot clé; touche <F>). Crée une boucle de répétition, pour que les lignes à l'intérieur de la boucle soient répétées autant de fois que spécifié. (La fin de la boucle est indiquée par une instruction NEXT.) Par exemple :

```
FOR I = 2 TO 20 STEP 2
```

La variable de contrôle dans cette instruction FOR est I. Pendant la répétition de la boucle, I sera incrémentée de 2 en 2 jusqu'à 20.

FUNCTION (majuscule; touche <NEW LINE>). Met l'ordinateur en mode fonction; la prochaine pression de touche sera prise en compte comme une touche "fonction".

GOSUB (mot clé; touche <H>). Donne le contrôle du programme à un sous-programme. Le numéro de la première ligne du sous-programme doit être fourni de façon littérale :

```
GOSUB 300
```

ou par un nom de variable numérique :

```
GOSUB T
```

ou même comme une expression numérique, pour avoir un appel "calculé" :

```
GOSUB 100 * (VAL I$ + 4)
```

GOTO (mot clé; touche <G>). Donne le contrôle du programme à n'importe quelle ligne spécifiée. Le numéro de ligne peut être un nombre littéral, une variable numérique, ou une expression numérique. (Voir GOSUB à titre d'exemple.)

GRAPHICS (majuscule; touche <9>). Met le clavier en mode graphique; les pressions de touches qui suivront seront prises en compte en tant que caractères graphiques, ou caractères en vidéo inversée. La touche GRAPHICS sert aussi à quitter le mode graphique, pour revenir dans le mode lettre.

IF (mot clé; touche <U>). Débute une instruction conditionnelle. Après le mot IF, doit se trouver une expression vraie ou fausse, puis la clause THEN; par exemple :

```
IF D < > 0 THEN LET Q = 1/D
```

Dans cette instruction, l'ordinateur commence par évaluer si D < > 0 est vraie ou est fausse. Si c'est "vrai", l'instruction LET (qui suit THEN) est exécutée; si c'est "faux", le contrôle du programme est directement donné à la ligne suivante.

**INKEY$** (fonction; touche <B>). A chaque occurrence de INKEY$, l'ordinateur scrute une fois tout le clavier pour voir si on est en train d'appuyer sur une touche. Si c'est le cas, INKEY$ fournit la valeur du caractère de la touche enfoncée; sinon INKEY$ fournit la chaîne de caractères "vide" ("").

INPUT (mot clé; touche <I>). Dit à l'ordinateur de lire une information depuis le clavier et de la ranger dans la variable spécifiée. Par exemple :

```
INPUT S$
```

lit une chaîne de caractères au clavier, et la stocke dans la variable de nom S$. INPUT affiche l'information introduite au clavier au fur et à mesure.

INT (fonction; touche <R>). Fournit la partie entière d'un nombre.

LEN (fonction; touche <K>). Fournit la longueur, en nombre de caractères d'une chaîne de caractères.

LET (mot clé; touche <L>). Affecte une valeur à une variable. La variable peut aussi bien être une nouvelle variable qui reçoit sa première valeur, qu'une ancienne variable qui reçoit une nouvelle valeur. A gauche du signe égal, l'instruction LET doit contenir un nom de variable unique; la partie droite peut comporter des littéraux, des variables ou des expressions. Par exemple :

```
LET V = 5
LET A = B
LET X = (5 * Y)/2
```

Toutes les variables se trouvant à droite du signe égal doivent avoir été définies avant l'appel de l'instruction LET. L'instruction LET ne modifie pas les valeurs des variables se trouvant à droite du signe égal.

LIST (mot clé; touche <K>). Affiche le programme actuel à l'écran. Si LIST est suivi d'un numéro de ligne, de cette façon :

```
LIST 100
```

la liste du programme commence à cette ligne.

LLIST (majuscule; touche <G>). Commande l'impression de la liste du programme sur l'imprimante.

LN (fonction; touche <Z>). Fournit le logarithme népérien (base e) d'un nombre positif.

LOAD (mot clé; touche <J>). Charge un programme rangé sur cassette en mémoire. La commande LOAD a deux formes : avec le nom du programme :

```
LOAD "Nom"
```

ou sans nom :

```
LOAD " "
```

Les guillemets sont obligatoires dans les deux formes. Dans la deuxième forme, l'ordinateur charge le premier programme qu'il rencontre sur la cassette.

LPRINT (majuscule; touche <S>). Ecrit sur l'imprimante une ligne d'information.

NEW (mot clé; touche <A>). Efface le programme actuel de la mémoire de l'ordinateur.

NEW LINE Introduit un ligne de programme ou une donnée dans la mémoire de l'ordinateur. Se trouve sur le ZX81.

NEXT (mot clé; touche <N>). Indique la fin d'une boucle FOR. La variable de contrôle, définie dans l'instruction FOR, doit être mentionnée après le mot NEXT; par exemple :

```
NEXT I
```

NOT (fonction; touche <N>). Modifie une expression vraie ou fausse à l'intérieur d'une instruction conditionnelle IF; par exemple :

```
IF NOT D > 0 THEN GOTO 10
```

Cette instruction est équivalente à :

```
IF D<=0 THEN GOTO 10
```

OR (majuscule; touche <W>). Crée une expression composée vraie ou fausse à l'intérieur d'une instruction conditionnelle IF. Dans l'exemple suivant, OR combine deux égalités :

```
IF X = 5 OR Y = 3 THEN GOSUB 300
```

L'instruction GOSUB qui se trouve après THEN ne sera exécutée que si au moins l'une des deux égalités est vraie.

PAUSE (mot clé; touche <M>). Indique à l'ordinateur d'interrompre l'exécution du programme. La durée de l'interruption est déterminée par le nombre entier qui suit le mot clé PAUSE. Par exemple :

```
PAUSE 50
```

provoque une interruption d'environ une seconde; en augmentant le nombre de multiples de 50, on fera durer l'interruption pendant autant de secondes supplémentaires. Si le nombre est supérieur à 32766, la commande signifie "s'arrêter définitivement". Toute interruption peut être annulée, ce qui aura pour effet de continuer l'exécution du programme, en appuyant sur l'une des touches du clavier.

PEEK (fonction; touche <O>). Fournit le contenu d'une position de la mémoire d'adresse comprise entre 0 et 65635.

PLOT (mot clé; touche <Q>). Affiche un "pixel" sur l'écran à une position donnée. Les éléments des coordonnées d'une position pour l'instruction PLOT sont dans l'ordre contraire à ceux d'une instruction PRINT AT. La forme générale de l'instruction PLOT est :

PLOT *h*,*v*

avec *h* coordonnée horizontale (comprise entre 0 et 63) et *v* coordonnée verticale (comprise entre 0 et 43). Le point de coordonnées (0,0), pour l'instruction PLOT, se trouve dans le coin en bas à gauche de l'écran.

POKE (mot clé; touche <O>). Ecrit une valeur à l'adresse mémoire spécifiée (comprise entre 0 et 65535). Prend la forme :

POKE *a,v*

où *a* est l'adresse d'une position de la mémoire, et *v* une valeur. La valeur absolue de *v* ne doit pas dépasser 255.

PRINT (mot clé; touche <P>). Affiche une ligne d'information à l'écran. Un point-virgule (;) sépare les différents éléments de l'instruction PRINT devant être affichés les uns à côté des autres, sans espace entre eux. Une virgule (,) entre les éléments provoque une tabulation vers le milieu de l'écran, ou au début de la ligne suivante.

RAND (mot clé; touche <T>). Initialise le point de départ d'une suite de nombres aléatoires produits par la fonction RND. Les instructions :

RAND

et

RAND 0

déclenchent un point de départ imprévisible. L'instruction :

RAND *n*

où *n* est compris entre 1 et 65536, provoque l'utilisation d'un point de départ connu.

REM (mot clé; touche <E>). Crée une ligne "commentaire" dans la liste du programme. L'ordinateur ignore les lignes REM à l'exécution du programme. On peut placer ce que l'on veut après le mot clé REM.

RETURN (mot clé; touche <Y>). Indique la fin d'un sous-programme. Redonne le contrôle du programme à la ligne qui suit l'instruction GOSUB d'où venait l'appel au sous-programme.

RND (fonction; touche <T>). Fournit un nombre "aléatoire" compris entre 0 et 1, bornes incluses.

RUBOUT (majuscule; touche <0>). Supprime le caractère, mot clé, ou fonction se trouvant immédiatement à gauche du curseur. RUBOUT peut être utilisé chaque fois qu'une ligne est en cours d'introduction dans l'ordinateur, ou quand une ligne complète est en train d'être éditée. Présente sur le ZX81.

RUN (mot clé; touche <R>). Commande à l'ordinateur de commencer l'exécution des lignes du programme actuellement en mémoire. (L'ordinateur commence par remettre à blanc toutes les variables résultats des travaux précédents.) Si RUN est suivi par un numéro de ligne, l'ordinateur commence l'exécution à la ligne indiquée.

SAVE (mot clé; touche <S>). Range un programme sur cassette. La commande SAVE nécessite un nom de programme, entre guillemets :

```
SAVE "Nom"
```

SCROLL (mot clé; touche <B>). Déplace toutes les informations affichées à l'écran d'une ligne vers le haut. La ligne du haut est perdue, et une ligne à blanc apparaît en bas de l'écran.

SGN (fonction; touche <F>). Fournit le *signe* d'un nombre quelconque; le résultat a pour valeur : –1, 0 ou 1.

SLOW (majuscule; touche <D>). Met l'ordinateur en mode "calcul lent".

SPACE Insère un caractère <espace> (ou <blanc>) dans une ligne de programme ou dans une chaîne de caractères.

SQR (fonction; touche <H>). Fournit la racine carrée d'un nombre positif ou nul.

STEP (majuscule; touche <E>). Dans l'instruction FOR, indique de combien augmentera (ou diminuera) la variable de contrôle à chaque répétition de la boucle. Par exemple :

```
FOR I = 5 TO 25 STEP 5
```

Dans cette instruction, la variable de contrôle I est incrémentée de 5 à chaque répétition de la boucle. Si la clause

STEP est omise dans une instruction FOR, le pas d'incrémentation pris par défaut sera 1.

STOP (majuscule; touche <A>). Arrête l'exécution d'un programme. Le mot STOP peut se trouver à l'intérieur d'un programme, ou être introduit au clavier, dans le but de terminer l'exécution du programme.

STR$ (fonction; touche <Y>). Fournit une chaîne de caractères équivalente à un nombre donné. Par exemple :

```
LET S$ = STR$ 1234
```

Cette instruction a pour résultat la création de la chaîne de caractères "1234" qui sera stockée dans la variable S$.

TAB (fonction; touche <P>). Utilisée à l'intérieur de l'instruction PRINT pour afficher une information à partir d'un numéro de colonne donné sur la ligne actuelle. Par exemple :

```
PRINT TAB 15; "BONJOUR"
```

écrira BONJOUR à partir de la colonne 15.

TAN (fonction; touche <E>). Fournit la tangente d'un nombre représentant un angle. TAN suppose que l'angle est donné en radians (1 degré = 0,0175 radian).

THEN (majuscule; touche <3>). Précède le résultat à l'intérieur d'une instruction IF. THEN doit toujours être suivi d'un mot clé. L'instruction se trouvant après THEN ne sera exécutée que si l'expression a été évaluée à "vraie".

TO (majuscule; touche <4>). Définit la valeur maximale que pourra prendre la variable de contrôle d'une boucle FOR. Par exemple :

```
FOR I = 1 TO 10
```

TO peut également être utilisée pour "la sélection" de sous-chaînes de caractères. L'instruction LET que voici affecte la valeur "ABC" à la sous-chaîne de caractères, incluse dans A$, commençant en position 2 et se terminant en position 4 :

```
LET S$(2 TO 4) = "ABC"
```

UNPLOT (mot clé; touche <W>). Efface un "pixel" de l'écran à une position donnée. La forme générale de l'instruction PLOT est :

UNPLOT *h,v*

avec *h* coordonnée horizontale (comprise entre 0 et 63) et *v* coordonnée verticale (comprise entre 0 et 43). Le point de coordonnées (0,0), pour l'instruction UNPLOT, se trouve dans le coin en bas à gauche de l'écran.

USR (fonction; touche <L>). Appelle un sous-programme en langage machine se trouvant à l'adresse mémoire indiquée; la forme générale de USR est :

USR *a*

où *a* est l'adresse en mémoire du point d'entrée d'un sous-programme.

VAL (fonction; touche <J>). Fournit la valeur numérique d'une chaîne de caractères. Les caractères peuvent être des chiffres, des fonctions, des calculs, ou des variables formant une expression numérique valide.

Appendice B

La codification des messages d'erreur du ZX81

A la fin de chaque exécution d'un programme, un message apparaît dans le coin en bas à gauche de l'écran sous la forme :

m/n

où *n* est le numéro de la ligne à laquelle l'exécution du programme s'est terminée, et *m* l'un des codes suivants :

0 Pas d'erreur; le programme s'est correctement terminé.

1 Non-concordance entre les variables de contrôle des instructions FOR et celles des instructions NEXT.

2 Utilisation d'une variable non définie.

3 L'indice d'un tableau sort des limites de l'intervalle défini dans l'instruction DIM.

4 L'ordinateur n'a plus assez de mémoire disponible.

5 Il n'y a plus de place sur l'écran.

6 Un calcul a donné un nombre trop grand pour pouvoir être manipulé par l'ordinateur.

7 Une instruction RETURN n'a pas été précédée d'une instruction GOSUB.

8 Une instruction INPUT a été utilisée en tant que commande en mode immédiat.

9 Programme terminé après une instruction STOP:

A Utilisation incorrecte d'une fonction (par exemple : SQR –1 ou LN 0).

B Adresse incorrecte, numéro ou entier incorrect dans une instruction telle que PRINT AT, PLOT, CHR$, etc.

C L'expression contenue dans la chaîne de caractères qui suit VAL ne peut être évaluée numériquement.

D Programme interrompu au clavier par l'une des touches BREAK ou STOP.

E Non utilisé.

F Commande SAVE non suivie d'un nom de programme.

INDEX

La bibliothèque SYBEX

INTRODUCTION AU PASCAL, Pierre Le Beux

500 pages, Réf. 222
Un livre complet et progressif pour l'apprentissage du Pascal. Depuis les concepts de base jusqu'aux programmes élaborés et les traitements de fichiers et graphiques.

LE PASCAL PAR LA PRATIQUE, Pierre Le Beux et Henri Tavernier

550 pages, Réf. 229
Plus de 140 exercices complètement traités : énoncé du problème, algorithmes et commentaires, programmes, exemples d'exécution.

LE GUIDE DU PASCAL, Jacques Tiberghien

450 pages, Réf. 232
Le dictionnaire encyclopédique complet du Pascal, définissant chaque mot, procédure, fonction des différentes versions du Pascal.

PROGRAMMES EN PASCAL pour Scientifiques et Ingénieurs, Alan R. Miller

350 pages, Réf. 240
Une présentation complète des algorithmes les plus fréquemment utilisés pour les applications scientifiques et techniques.

JEUX EN PASCAL SUR APPLE, Douglas Hergert et Joseph T. Kalash

350 pages, Réf. 241
Un ensemble des jeux les plus populaires en Pascal UCSD, invitant le lecteur non seulement à jouer mais à comprendre la manière dont les jeux sont exécutés sur l'ordinateur.

INTRODUCTION AU BASIC, Pierre Le Beux

336 pages, Réf. 216
Une introduction progressive au BASIC, couvrant tous les aspects du langage actuellement disponibles sur microordinateurs et systèmes de temps partagé.

LE BASIC PAR LA PRATIQUE : 60 exercices, Jean-Pierre Lamoitier

250 pages, Réf. 231, 2e édition
Les exercices, complètement traités (énoncé, analyse, solution avec ordinogramme, programme d'application) permettent au lecteur de perfectionner très rapidement ses connaissances en BASIC. Tous les programmes sont en BASIC Microsoft.

AU CŒUR DES JEUX EN BASIC, Richard Mateosian
300 pages, Réf. 233
Enseigne la programmation du BASIC par les jeux. Ces jeux sont en BASIC Microsoft et peuvent être utilisés sur TRS80, APPLE II et PET/CBM.

LE BASIC POUR L'ENTREPRISE, Xuan Tung Bui
172 pages, Réf. 253, 2e édition
Ce livre présente toutes les méthodes importantes de gestion en expliquant leur but, leur principe ainsi que leur réalisation en BASIC. Il est constitué d'un choix de programmes testés, et prêts à être utilisés pour la décision d'entreprise.

JEUX D'ORDINATEUR EN BASIC, David H. Ahl
192 pages, Réf. 246
100 jeux passionnants pour jouer avec votre ordinateur personnel, seul ou à plusieurs. Pour chaque jeu, programme et exemple d'exécution.

NOUVEAUX JEUX D'ORDINATEUR EN BASIC, David H. Ahl
204 pages, Réf. 247
Complément indispensable de *Jeux d'ordinateur en BASIC,* 84 jeux supplémentaires, faciles à utiliser sur tout microordinateur.

PROGRAMMES EN BASIC pour Scientifiques et Ingénieurs, Alan R. Miller
315 pages, Réf. 259
Second volume de la série SYBEX « Programmes pour Scientifiques et Ingénieurs ». Un jeu complet des algorithmes scientifiques les plus importants, et leur implémentation en BASIC.

PROGRAMMEZ EN BASIC SUR TRS-80, Léopold Laurent
T. 1, 208 pages, Réf. 250; T. 2, 304 pages, Réf. 251
Une initiation au langage BASIC et à la programmation comprenant de nombreuses applications pratiques : facturation automatisée, traitement de texte, gestion de fichiers de données...

INTRODUCTION À ADA, Pierre Le Beux
350 pages, Réf. 242
Un ouvrage d'introduction au nouveau langage ADA, destiné aux étudiants, aux professionnels et aux ingénieurs. Ce livre aborde les concepts très progressivement et fournit de nombreux exemples de programmes.

VOTRE PREMIER ORDINATEUR, Rodnay Zaks
290 pages, Réf. 226
Une introduction simple et complète aux petits ordinateurs avec leurs périphériques. Ce qu'ils peuvent faire et comment se les procurer.

PROGRAMMATION DU Z80, Rodnay Zaks
600 pages, Réf. 220
Une introduction complète à la programmation en langage assembleur pour le Z80.

GUIDE DU CP/M AVEC MP/M, Rodnay Zaks
340 pages, Réf. 228
Un guide de référence indispensable sur le CP/M, système d'exploitation standard le plus utilisé sur les microordinateurs.

INTRODUCTION AU TRAITEMENT DE TEXTE, Hal Glatzer

250 pages, Réf. 243
Ce livre vous apprendra ce que le traitement de texte peut faire pour vous. Quel type de matériel est le meilleur. Comment écrire et éditer avec le traitement de texte.

INTRODUCTION À WORDSTAR, Arthur Naiman

200 pages, Réf. 255
Apprenez facilement à utiliser le WordStar, un programme de traitement de texte hautement performant pour tout utilisateur d'ordinateur.

PROGRAMMATION DU 6502, Rodnay Zaks

300 pages, Réf. 224, 2e édition
La programmation des systèmes basés sur le microprocesseur 6502, des concepts de base aux structures de données les plus avancées.

APPLICATIONS DU 6502, Rodnay Zaks

300 pages, Réf. 219
Techniques d'applications pratiques : le livre des entrées-sorties du 6502.

PROGRAMMATION DU 6800, Daniel-Jean David et Rodnay Zaks

380 pages, Réf. 218
Pour apprendre à programmer le 6800, avec de nombreux exercices. Un véritable apprentissage par l'action.

LEXIQUE INTERNATIONAL MICROPROCESSEURS, avec dictionnaire abrégé en 10 langues

192 pages, Réf. 234
Lexique de poche contenant toutes les définitions et les sigles du langage informatique avec, en plus, un dictionnaire abrégé des principaux termes en 10 langues.

DU COMPOSANT AU SYSTÈME : une introduction aux microprocesseurs, Rodnay Zaks

620 pages, Réf. 239
Tous les concepts et techniques liés aux microprocesseurs, depuis les principes de base jusqu'à la programmation. Une introduction simple et complète.

TECHNIQUES D'INTERFACE, Austin Lesea et Rodnay Zaks

450 pages, Réf. 230, 3e édition
Exposé clair et systématique sur les méthodes et les composants qui permettent de construire un système complet.

VOTRE ORDINATEUR ET VOUS, Rodnay Zaks

238 pages, Réf. 271
La plupart des « pannes d'ordinateur » sont causées par l'ignorance ou la négligence. Voici le premier ouvrage qui explique comment éliminer tous les problèmes d'ordinateur. Pour tous ceux qui possèdent ou envisagent l'achat d'un microordinateur.

INTRODUCTION AU p-SYSTEM UCSD, Charles W. Grant et Jon Butah
250 pages, Réf. 257
Une introduction simple et claire à l'utilisation du p-SYSTEM UCSD.

PROGRAMMEZ EN BASIC SUR VIC 20, G.O. Hamann
2 tomes, Réf. 244-245

PROGRAMMEZ VOS JEUX SUR GOUPIL, F. Abella
200 pages, Réf. 264

DÉCOUVREZ LE SHARP PC-1500 ET LE TRS-80 PC-2, M. Lhoir
2 tomes, Réf. 261-262

PROGRAMMEZ EN BASIC SUR APPLE II, Léopold Laurent
208 pages, tome 1, Réf. 268

EXERCICES EN BASIC SUR L'ORDINATEUR PERSONNEL IBM, Jean-Pierre Lamoitier
242 pages, Réf. 267

La plupart de ces ouvrages existent en version anglaise. N'hésitez pas à demander notre catalogue.

EN ANGLAIS

BASIC EXERCISES FOR THE APPLE
Jean-Pierre Lamoitier 232 pages, Réf. 0-084

BASIC FOR BUSINESS
Douglas Hergert 224 pages, Réf. 0-080

CELESTIAL BASIC : Astronomy on your Computer
Eric Burgess 228 pages, Réf. 0-087

INTRODUCTION TO PASCAL (Including UCSD Pascal)
Rodnay Zaks 422 pages, Réf. 0-066

DOING BUSINESS WITH PASCAL
Richard Hergert, Douglas Hergert 380 pages, Réf. 0-091

MASTERING VISICALC
Douglas Hergert 224 pages, Réf. 0-090

DOING BUSINESS WITH VISICALC
Stanley R. Trost 260 pages, Réf. 0-086

POUR UN CATALOGUE COMPLET DE NOS PUBLICATIONS

EUROPE
4, place Félix-Éboué
75583 Paris Cedex 12
Tél. : (1) 347.30.20
Telex : 211801

U.S.A.
2344 Sixth Street
Berkeley, CA 94710
Tel. : (415) 848.8233
Telex : 336311

ALLEMAGNE
Heyestr. 22
4000 Düsseldorf 12
Tel. : (0211) 287066
Telex : 08588163

SYBEX

Soulisse et Cassegrain, Imprimeurs, Niort (Deux-Sèvres).
Dépôt légal : Juin 1983. N° 2093.

DECOUVREZ LE ZX 81 et le Timex Sinclair 1000

Cet ouvrage vous expliquera, dans un langage clair et simple, comment utiliser toutes les possibilités de votre ZX81 ou de votre TIMEX SINCLAIR 1000.

Vous apprendrez rapidement comment :

— brancher votre téléviseur et votre magnétophone à cassettes à l'ordinateur et les faire travailler ensemble,

— utiliser le clavier pour donner des ordres à votre ordinateur,

— écrire vos propres programmes de dessin, de calcul, de jeux...

— utiliser le langage de programmation BASIC,

— utiliser les programmes « tout prêts » de ce livre. Ils sont destinés à :

- transformer votre microordinateur en super calculateur,
- tracer des histogrammes pour le calcul des finances familiales,
- dessiner sur votre écran.

l'Auteur

Douglas Hergert fait partie de la direction éditoriale de SYBEX Inc. Il est l'auteur de *BASIC for Business* et de *Mastering VisiCalc*. Il a apporté sa propre expérience des ordinateurs en collaborant à plusieurs ouvrages de SYBEX en tant qu'éditeur et traducteur.

ISBN 2-902414-56-0

256 6-83 79 F

DECOUVREZ LE

ZX 81 et le Timex Sinclair 1000

SYBEX

AMSTRAD CPC



MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

ACME https://acpc.me/